ハイスピードＲＩＳＣライクプロセッサー

ＰＩＣ１６Ｃ８４ ＥＥＰＲＯＭ内蔵で書き替え回数１００万回！

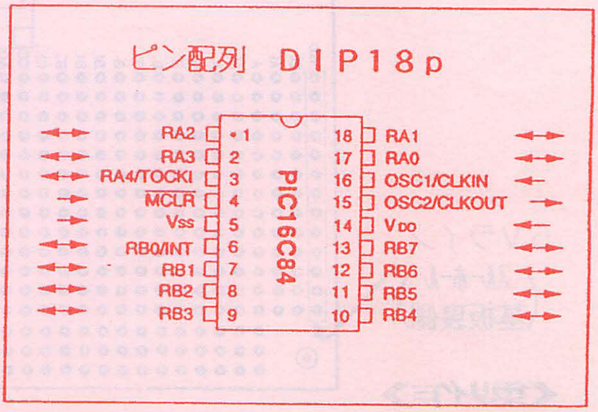
マイコンボード

ＡＫＩ（アキ）ＰＩＣ（ピック）プラチナキット（10MHz）

《ＰＩＣ１６Ｃ８４の主な特徴》

☆ワンチップ高性能ＲＩＳＣライクプロセッサ　ＰＩＣシリーズ使用
☆命令数は３５命令で、シングルワード構成
☆命令はシングルサイクル（400ns）で実行　10MHz時
　ただし、ブランチ命令のみ２サイクル
☆PICチップに　EEPROM（プログラムメモリ）　1024×14ビット
　　　　　　　　SRAM　36×8ビット
　　　　　　　　EEPROM（データメモリ）　64×8ビット　を内蔵
☆内蔵ＥＥＰＲＯＭは1,000,000回書き替え可能

☆Ｉ／Ｏポートは全部で１３個。
　吸い込み電流25mA max
　吐き出し電流20mA max
　LEDが直接ドライブできます。
☆８ビット クロック／カウンタ
　プログラマブルプリスケラー
☆４種の割り込みモード
☆パワーオンリセット回路内蔵



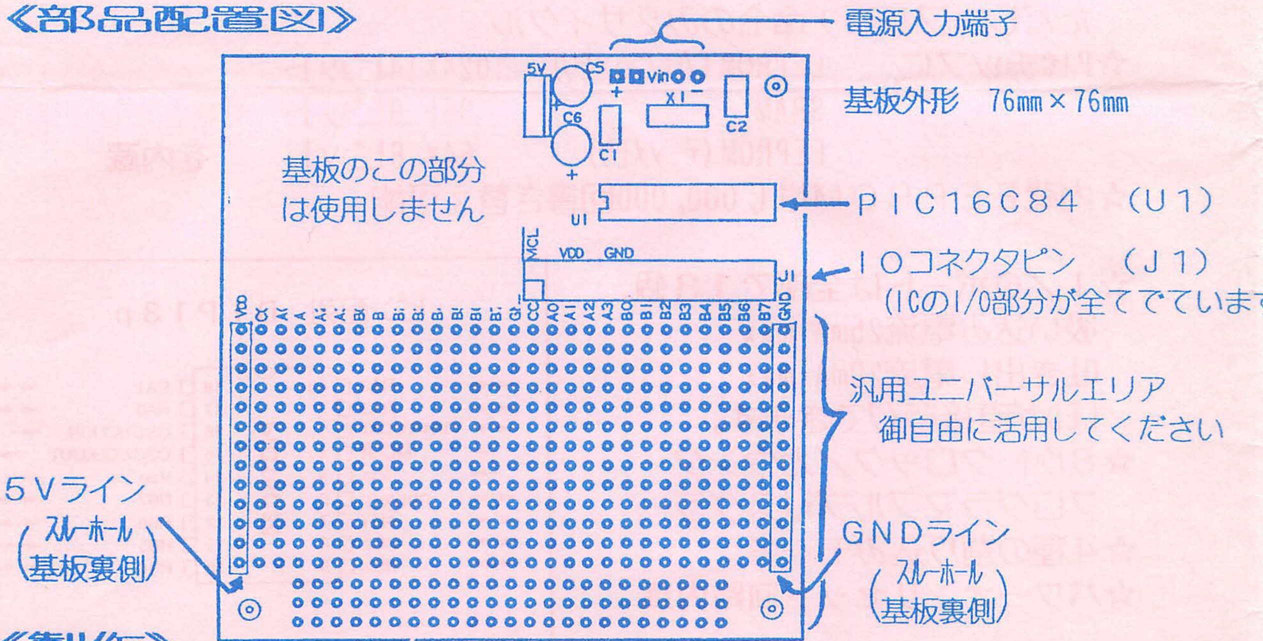
《キット概要》

□高品質な両面スルーホール専用基板を使用
□電源回路、発振回路はすべて専用パターンになっているため部品を半田付けするだけでＯＫです。
□ＩＣのＩ／Ｏはすべて、コネクタエリアに出ています。
□16×29グリッドのユニバーサルエリアを設けてあるため小さなシステムならばこのキットのみで製作ができます。

《パーツリスト》

部品名	数量	シルク記号	備考
専用ボード　ＡＫＩ－ＰＩＣ	1		
ＣＰＵ　PIC16C84-10/P	1	Ｕ１	ＥＥＰＲＯＭ内蔵型　１０ＭＨｚ動作版
７８０５　定電圧ﾚｷﾞｭﾚｰﾀIC	1	５Ｖ	2930-5, 2940-5の場合あり
水晶　１０ＭＨｚ	1	Ｘ１	
コンデンサ　10μＦ～100μＦ 0.1μＦ～1.5μＦ 15ｐＦ～47ｐＦ	1 1 2	Ｃ５ Ｃ６ C1, C2	10Ｖ以上　電解コンデンサ 表示（104～155）積層ｾﾗﾐｯｸｺﾝﾃﾞﾝｻ 表示（15～47）ｾﾗﾐｯｸｺﾝﾃﾞﾝｻ
１８ピンＩＣソケット	1		ＰＩＣ１６Ｃ８４用
９Ｖ電池スナップ	1		

《部品配置図》



《製作》

◇まずこの取説を一通り読んでください。

①ICソケットを取り付けます。ICソケットは切り欠きを基板のシルク印刷にあわせて取り付けてください。

②電源回路の定電圧レギュレータIC, コンデンサを取り付けます。ICの向きは、基板のシルク印刷の向きにあわせて下さい。外部から安定した4.5～5.5Vを加えられる場合は、定電圧レギュレータICの出力と、入力の穴を接続して下さい。

One Chip RISC-like Processor

PICシリーズ用プリント基板 NEWタイプ

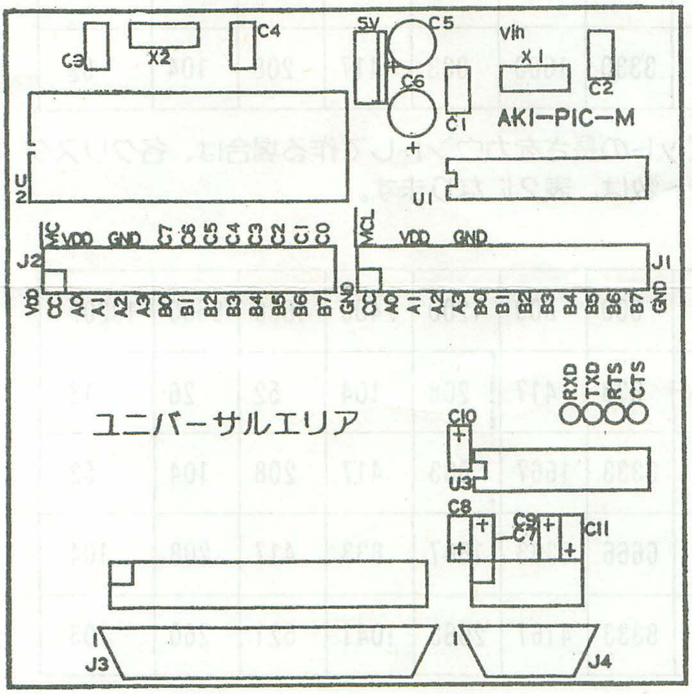
★ワンチップRISCライクプロセッサPICシリーズ用プリント基板が、新しく使いやすくなりました。

★RS232通信ができるようにAD232（MAX232C上位コンパチ版）・コンデンサ・Dsubコネクタが付けられるようになりました。また、パラレル通信用にDsub25Pコネクタが付けられるようになりました。

★PICチップ・電源・I/Oコネクタは、全て専用パターンになっています。

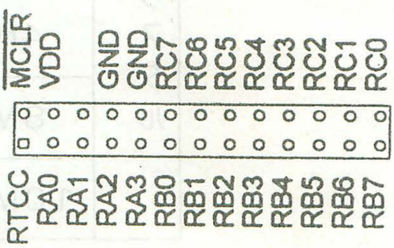
★ユニバーサルエリアを設けてあるのでマイコン制御機器の開発・製作に最適です。

《図１：部品配置図》



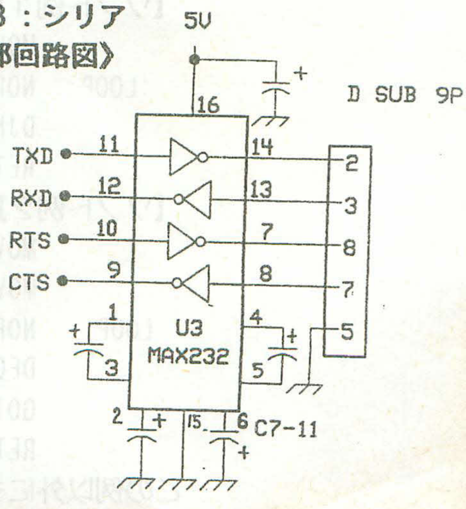
U1　：PIC16C54, 56, 71, 84
U2　：PIC16C55, 57
U3　：AD232 レベルコンバータIC
X1, X2　：水晶振動子またはセラミック発振子
C1~C4　：発振用コンデンサ（5～22pF）
C5, C6　：電源用コンデンサ（0.1～33μF）
C7~C11：AD232用外付コンデンサ（1μF）
J1, J2　：PIC-IC 入出力ピンコネクタ端子26P
J3　：パラレル用Dsub25Pコネクタ
J4　：シリアル用Dsub9Pコネクタ

《図２：J１．J２コネクタピン配置》



◆J４のピン配置は、モデム等と同じDCE側（回線終端装置側）になっています。
コンピュータ等と同じDTE側（データ端末装置側）でご使用になる場合は、基板－コネクタ間の配線をクロスしてご使用ください。

《図３：シリアル部回路図》



《AKIプラチナキットでは次の部品が追加されます》

U3	AD232
C7～C11	コンデンサ1μF×5個
J3	Dsub25Pコネクタ
J4	Dsub9Pコネクタ

ＲＳ２３２シリアル通信のビットレート作成について

【信号例】



通信速度３００～１９２００(BIT/SEC)の場合の１ビットの長さ(μS)は、表１になります。

【表１　１ビットの長さ】

通信速度(BIT/SEC)	300	600	1200	2400	4800	9600	19200
１ビットの長さ(μS)	3330	1660	833	417	208	104	52

ＰＩＣシリーズで１ビットの長さをカウントして作る場合は、各クリスタルによる必要なカウント数は、表２になります。

【表２　カウント数】

通信速度(BIT/SEC)		300	600	1200	2400	4800	9600	19200
クリスタル	１ＭＨｚ	833	417	208	104	52	26	13
	４ＭＨｚ	3333	1667	833	417	208	104	52
	８ＭＨｚ	6666	3333	1667	833	417	208	104
	１０ＭＨｚ	8333	4167	2083	1041	521	260	103

【ソフト例１】

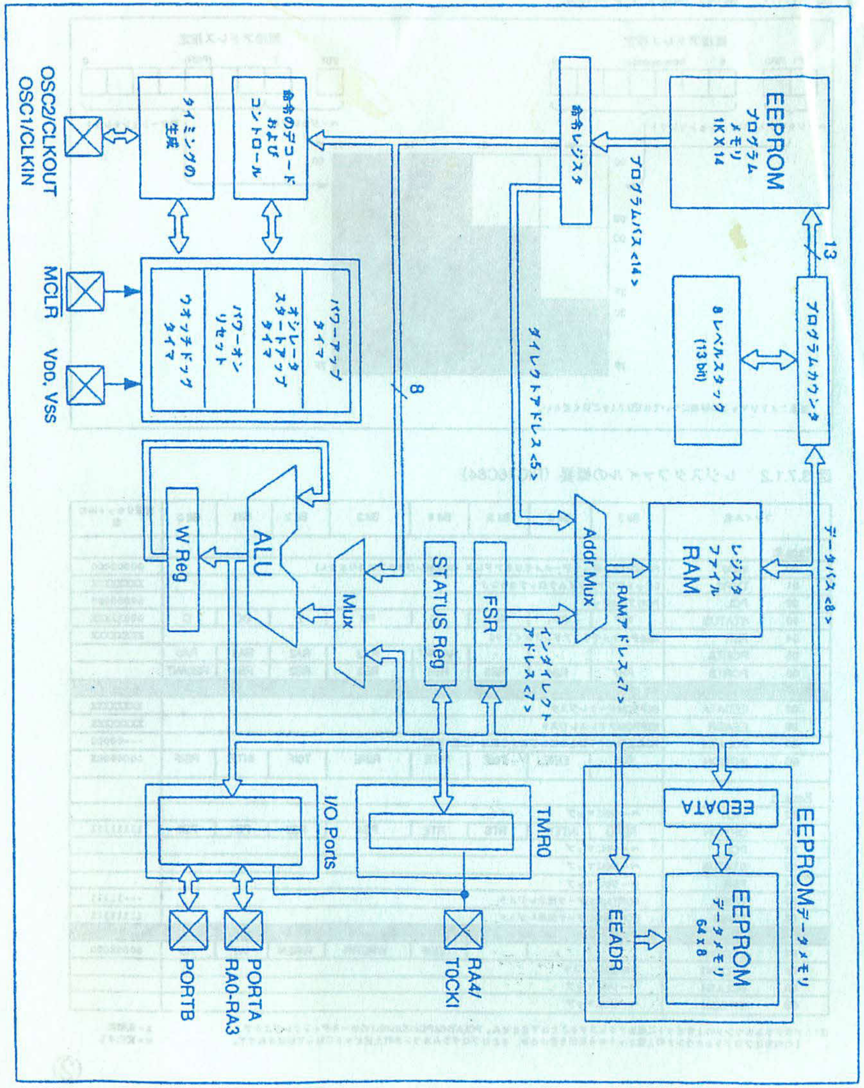
```
        MOV     RS_CUNT,#BIT_L      :10MHz 9600の場合  260÷4=65
:LOOP   NOP                         :ＮＯＰの数で調整する。
        DJNZ    RS_CUNT,:LOOP
        RET
```

【ソフト例２】

```
        MOVLW   BIT_L               :10MHz 9600の場合  260÷4=65
        MOVFW   RS_CUNT
LOOP    NOP                         :ＮＯＰの数で調整する。
        DECFZ   RS_CUNT,1
        GOTO    LOOP
        RET
```

この例以外にもＲＴＣＣを使用する方法もあります。

図 B　　PIC16C84 ブロック図



1.0 はじめに

PIC16C84は、低価格、高性能、CMOS、完全スタティック、EEPROMベースの8ビットマイクロコントローラです。EEPROMプログラムメモリは、コード開発だけでなく、完全生産用のワン・タイム・プログラマブルメモリとしても使用できるように設計されています。このプログラムメモリはコード実行中には更新できません。しかし、わずか2つのピンを使用するだけでデータの逐次入出力を可能とする特殊な"インシステムプログラミング"機能が用意されていますから、ユーザはシステムに埋め込まれているPIC16C84のプログラムコードを更新することができます。EEPROMデータメモリ(64バイト)のリードとライトは、全VDD範囲(2.0V-6.0V)で、実行できます。

PIC16C84は最新のRISCに似たアーキテクチャを採用しています。35個の縮小命令セット、シングルワードの命令語(14ビット幅)、シングルサイクルの命令実行(プログラム分岐だけは2サイクル)、命令のパイプライン実行、大きなレジスタセット、命令用とデータ用のそれぞれ独立したメモリ(ハーバードアーキテクチャ)構成など、革新的なアーキテクチャの採用により、PIC16C84は非常に高い性能を実現することができました。同クラスの他の8ビットマイクロコントローラと比べて、PIC16C84はコード圧縮で2:1、スピードで4:1の性能向上を実現しています。

PIC16C84は外部部品を少なくしシステムコストを押さえ、システムの信頼性を高め電力消費を少なくするなどの特長を備えています。4種類のオシレータオプションがあり、1本のピンによるRCオシレータによって低価格の設計ができ、LPオシレータによって消費電力を押さえることができます。SLEEP(パワーダウン)モードによってさらに消費電力を押さえることができます。外部割り込みとリセットによってSLEEPモードからチップを動作状態に戻すことができます。

専用のRCオシレータ内蔵の高信頼性のウオッチドッグタイマによって、ソフトウエアの誤動作に対して保護することができます。

1.1 PIC16C5Xとの互換性

マイクロコントローラのPIC16C5Xファミリをご存じの方は、これがPIC16C5Xアーキテクチャの改良版であることをご理解いただけると思います。詳細については付録Aの改良点の一覧を参照してください。PIC16C5X用に書かれたコードはPIC16C84に簡単に移植できます(付録B参照)。

1.2 アプリケーション

PIC16C84は高速車載用、機器のコントロールから、小電力リモートセンサ、電子錠、保安機器までのアプリケーション範囲に利用できます。また、PIC16C84はスマート・カードやRFタグ用のマイクロコントローラとしても理想的です。EEPROMテクノロジによってアプリケーションプログラム(トランスミッタコード、モータスピード、レシーバ周波数など)のカスタマイズが短期間に最適にできます。スルーホールや表面実装用の小型端子パッケージを使用することよって、占有面積に制限のあるアプリケーションすべてに対して、このマイクロコントローラシリーズが最適です。低価格、低消費電力、高性能、使いやすさ、I/Oのフレキシビリティによって PIC16C84 は今まで検討されていてまだマイクロプロセッサが使われていない分野(タイマ機能、大型システムの周辺ロジック、コプロセッサのアプリケーションなど)にも大いに機能を発揮できます。

さらに、PIC16C84のインシステムプログラミング機能(2つのピンを使ってデータ転送を行います)を利用すれば、組立てや試験が完了した後に製品をカスタマイズすることができます。

この機能を利用すると、製品をシリアル化したり、最終試験実施後にしか入手できない校正データを保存したり、完成品のファームウェアをアップグレードしたりすることができます。

2.0 PIC16C84 デバイス構成

いろいろな周波数範囲、パッケージオプションがあります。アプリケーションと量産の必要に応じて、このセクションの説明と表を使って適当なデバイスオプションを選択できます。ご注文の時は、このデータシートの最後のページにあるPIC16C84 製品番号指定表を使って正しい製品番号をご指定ください。

2.1 クイックターンアラウンドプロダクション(QTP)デバイス

Microchip 社では工場での量産用 QTP プログラムサービスのご注文をお受けいたします。このサービスはデバイスの大量の書き込み負担を低減する事を目的とし、しかもそのコードパターンが安定したものになったユーザに対して用意されています。工場で、EEPROMプログラムメモリにすべてのロケーションとヒューズオプションをプログラムした上でお手元にお届けします。量産出荷が行われる前に実際のコードとプロトタイプの確認を行います。詳細については Microchip Technology の販売店にお問い合わせください。

2.2 シリアルクイックターンアラウンドプロダクション(SQTP)デバイス

Microchip社は、各デバイス中の少数のロケーションに他と異なるシリアル番号をプログラムするというユニークなプログラミング・サービスを行っています。割り当てる番号は、ランダム番号、疑似ランダム番号、順次番号のどれでもかまいません。

シリアル・プログラミングでは、エントリコード、パスワード、またはID番号として使用できる一意の番号を各デバイスに割り当てることができます。

①

3.5 プログラムカウンタモジュール

プログラムカウンタ(PC)は13ビット幅です。その下位バイト、PCLはリード、ライト可能なレジスタです。PCの上位バイト、PCHは直接リード、ライトできません。PCの上位バイトはPCLATH(0Ah)を通してライトされます。CALL、GOTOまたはPCLへのライトの間にPCに新しい値がロードされたときは、図3.5.1に示すようにPCLATHからPCの上位ビットにロードされます。

3.6 スタック

PIC16C84には8レベル×13ビット幅のハードウエアスタックがあります。スタックのスペースはプログラムあるいはデータスペースの一部ではなくスタックポインタはリード、ライトはできません。PC は CALL 命令が実行されたとき、または割込みが認識されたとき、スタックにプッシュされます。そのスタックはRETURN、RETLW、RETFIE 命令の実行されたときポップされます。PCLATH(0Ah)はプッシュあるいはポップ動作で影響を受けません。

3.7 レジスタファイル構成

PIC16C84のレジスタファイルは128×8で構成されており、ファイルセレクトレジスタFSRを通して直接または間接にアクセスされ、データメモリとしても参照されます。STATUSレジスタにレジスタファイルのページセレクタビットがあり、4ページまで選択できます。しかし、PIC16C84のデータメモリは2Fhまでしか内臓していません。最初の12ロケーションは特殊用途レジスタを配置するのに使われます。ロケーション0Ch-2Fhはスタティック RAM で実現されている汎用レジスタです。複数の特殊用途レジスタはページ 1に配置されます。ページ 1の時に、ロケーション8Ch-AFhをアクセスするとページ 0 の RAMをアクセスすることになります(図 3.7.1)。

3.7.1 レジスタファイルのアドレス指定モード

レジスタファイルに対しては、直接的または間節的なアドレス指定が可能です。どちらのモードでも、最大512個のレジスタロケーションにアドレスを指定できます。

直接アドレス指定モード： 図3.7.1.1に示す通り、OPCODEからの直接アドレス7ビットとステータスレジスタからの2ビット（RP1、RP0）を連結させることによって、有効な9ビットの直接アドレスを作成することができます。

間接アドレス指定モード： 上記の他に、ファイルアドレス00h（INDF）を使用して間接的にアドレスを指定することもできます。ファイルレジスタとしてINDFを使用したすべての命令は、実際には、ファイルセレクトレジスタ（FSR）が指示するファイルをアクセスすることになります。INDF自体を間接的にリードすると00hが読み出されます。INDFに間接的なライトを実行しても、（ステータスビットが変化するだけで）書込みは行われません。FSRレジスタの8ビットとステータスレジスタのIRPビットを連結させることによって、有効な9ビットのアドレスを作成することができます。

特殊機能をもつレジスタの中にはページ1にマップされているものがあります。したがって、RP0ビットをセットしてこれらをアドレス指定する必要があります。RP1ビットとIRPビットは実質的には使用されません。

使いやすさを考えて、汎用レジスタはページ0とページ1の両方にマップしてあります。

図 3.5.1 異なった条件でのロードPC

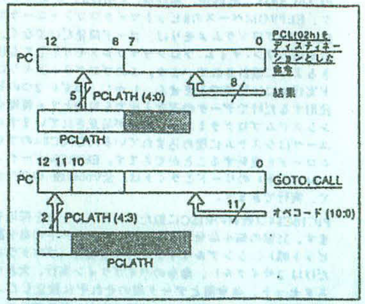


図 3.7.1 レジスタファイルマップ

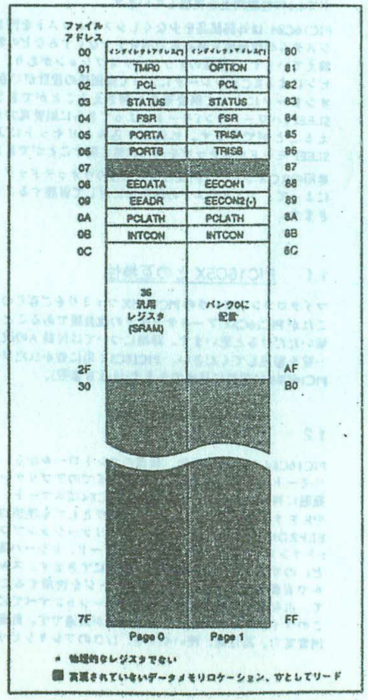


図 3.7.1.1 直接／間接アドレス指定

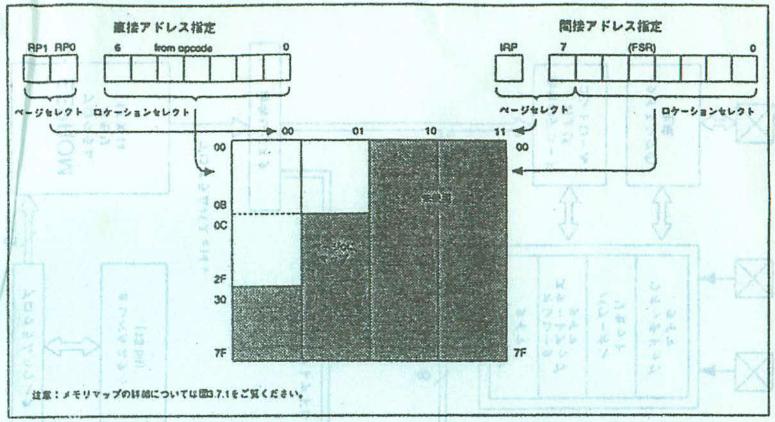


図 3.7.1.2 レジスタファイルの概要（PIC16C84）

ファイル名		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit1	Bit 0	電源リセット時の値
Page 0:										
00	IND0	FSRの内容を使ってデータメモリをアドレス（物理的レジスタではありません）								00000000
01	TMR0	8ビットのリアルタイムクロックカウンタ								XXXXXXXX
02	PCL	PCの下位8ビット								00000000
03	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	00011XXX
04	FSR	間接データメモリ、アドレスポインタ0								XXXXXXXX
05	PORTA	-	-	-	RA4/RT	RA3	RA2	RA1	RA0	
06	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	
07		未実装								
08	EEDATA	EEPROMデータレジスタ								XXXXXXXX
09	EEADR	EEPROMアドレスレジスタ								XXXXXXXX
0A	PCLATH	PCの上位バイトのためにレジスタを保持（注意1参照）								---00000
0B	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000000X
Page 1:										
80	IND0	ページ0にマップ								
81	OPTION	RBPU	INTEDG	RTS	RTE	PSA	PS2	PS1	PS0	11111111
82	PCL	ページ0にマップ								
83	STATUS	ページ0にマップ								
84	FSR	ページ0にマップ								
85	TRISA	PORTA(105)データ指示レジスタ								---11111
86	TRISB	PORTB(106)データ指示レジスタ								11111111
		未実装								
88	EECON1	-	-	-	EEIF	WRERR	WREN	WR	RD	0000X000
89	EECON2	非物理的レジスタ								
8A	PCLATH	ページ0にマップ								
8B	INTCON	ページ0にマップ								

注1：プログラムカウンタの上位ビットに直接アクセスすることはできません。PCLATHはPC<15:8>向けのホールディングレジスタで、その内容はプログラムカウンタの上位ビットから更新を受けるか、またはプログラムカウンタの上位ビットに対して伝達されます。

x = 未知数
u = 変化なし

②

4.0 CPUの特長

マイクロコントローラを他のプロセッサと区別している特長は、リアルタイムの応用で必要な機能を実現する特定の回路です。PIC16C84にはシステムの信頼性を高め、外部部品を減らしてコストを低減し、動作消費電力の節減ができ、コードプロテクションができるなどの特長を実現する機能があります。

PIC16C84にはEEPROMのヒューズによってのみ停止の状態にできるオンチップのウオッチドッグタイマがあり、専用のRCオシレータで信頼性を高めています。パワーアップ時に必要な遅延時間をつくるために2個のタイマがあります。ひとつはオシレータスタートアップタイマ(OST)で、クリスタルオシレータが安定するまでチップをリセット状態に保ちます。もうひとつはパワーアップタイマ(PWRT)で、パワーアップ時にのみ 72ms (平均) の一定遅延時間をつくり、電源が安定する間チップをリセット状態に保つよう設計されています。チップに内蔵されたこの2個のタイマによって、ほとんどの応用で外部リセット回路がいらなくなります。

非常に消費電力の少ないパワーダウンモードを実現するために設計されたSLEEPモードがあります。SLEEPモードからは外部リセット、ウオッチドッグタイマのタイムアウト、割込みのどれかによって 動作モードに戻ることができます。複数のタイマオプションによってもこのチップをいろいろな応用に合わせることができます。RCオシレータオプションによってシステムコストを削減することができ、LPクリスタルオプションによって消費電力を節約することができます。EEPROMコンフィギュレーションビット(ヒューズ)の組み合わせによって色々なオプションを利用することができます(4.6章)。

4.1 リセット

PIC16C84には複数の種類の異なったリセットがあります:

a) パワーオンリセット(POR)

b) 通常動作時の$\overline{MCLR}$リセット

c) SLEEPモード時の$\overline{MCLR}$リセット

d) 通常動作時のWDTタイムアウトリセット

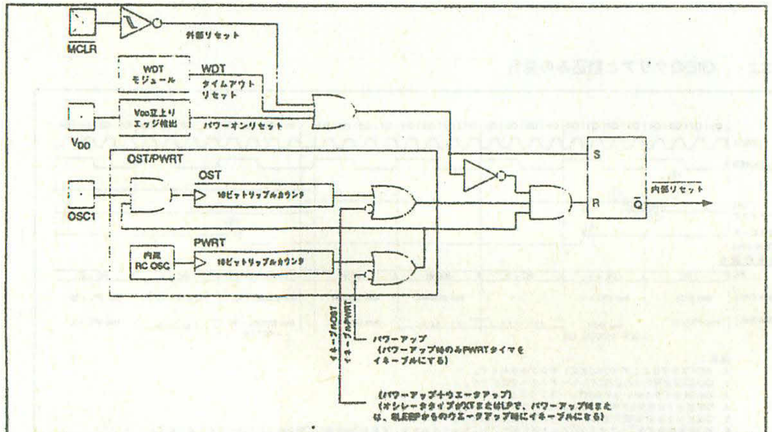
e) SLEEPモード時のWDTタイムアウトリセット

リセットされないレジスタがいくつかあります:PORでは不定、他のリセットでは変化が起こりません。ほかのほとんどのレジスタはパワーオンリセット(POR)、通常動作時の$\overline{MCLR}$またはWDTリセット、SLEEPモード時の$\overline{MCLR}$リセットによってリセット状態になります。SLEEPモード中の WDTリセットが通常動作の再開を実現させるので、これによってこのレジスタは影響を与えられません。この場合少し例外があります。PCは常にすべて0(0000h)にリセットされ、最後に、$\overline{TO}$と$\overline{PD}$ビットが3.9.2章で説明したようにそれぞれ異なったリセット環境でセットまたはリセットされます。リセットの性質を見るためにソフトウエアでこのビットを使います。すべてのレジスタのリセット状態の説明については表4.1.1をご覧下さい。(汎用レジスタ領域はパワーオンリセットによってクリアーされませんので、ユーザがソフトウエアによってクリアする必要があります。)

4.2 パワーオンリセット(POR)、パワーアップタイマ(PWRT)、オシレータスタートアップタイマ(OST)

<u>パワーオンリセット(POR)</u>: VDDの立ち上がり(1.2Vから1.8Vの範囲) が検出されると、パワーオンリセットパルスが内部で発生します。PORの利点を使うには、単に$\overline{MCLR}$ピンを直接(または抵抗を通して)VDDに接続します。これによって通常パワーオンリセットを発生するために必要になる外部のRC部品がいらなくなります。

図4.0.1　内部リセット回路の等価ブロック図



 Preliminary

表4.1.1　レジスタのリセット状態

レジスタ	アドレス	パワーオンリセット	通常動作時のWDTタイムアウトリセット	SLEEPモード時のWDTタイムアウトリセット	通常動作時の$\overline{MCLR}$リセット	SLEEPモード時の$\overline{MCLR}$リセット	割込みによるウェークアップ時
W	-	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
INDIR	00h	-	-	-	-	-	-
TMR0	01h	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
PC	02h	0000h	0000h	PC + 1	0000h	0000h	PC + 1
STATUS	03h	0001 1xxx	0000 1uuu	uuu0 0uuu	000u uuuu	0001 0uuu	uuu1 0uuu
FSR	04h	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
PORT A	05h	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
PORT B	06h	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TRIS A	85h	---1 1111	---1 1111	---u uuuu	---1 1111	---1 1111	---u uuuu
TRIS B	86h	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111	uuuu uuuu
OPTION	81h	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111	uuuu uuuu
EEDATA	08h	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
EEADR	09h	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
EECON1	88h	---0 0000	---0 ?000†	---u uuuu	---0 ?000†	---0 ?000†	---u uuuu
EECON2	89h	-	-	-	-	-	-
PCLATH	0Ah	---0 0000	---0 0000	---u uuuu	---0 0000	---0 0000	---u uuuu
INTCON	0Bh	0000 000x	0000 000u	uuuu uuuu	0000 000u	0000 0000	uuuuuuuu*

記号:　- = 影響なし、0をリードする　u = 変化なし　x = 不定

* 割込みによるウエークアップ動作時は1個またはそれ以上のフラグがセットされる。INTCONの他のビットは変化しない。

† EEPROMライト中にリセットがあった場合には、WRERR (3ビット) がセットされます。

VDDが低下するとき(または電圧低下の途中)には、POR回路から内部リセットは発生しません。

<u>パワーアップタイマ(PWRT)</u>: パワーアップタイマはパワーアップ時のみPORから72msの一定タイムアウト時間を発生します。このパワーアップタイマは内部のRCオシレータによって動作します。このチップはPWRTが動作中の間リセット状態を保ちます。PWRTによる遅延によってVDDが必要なレベルまで立ち上がることができます。コンフィギュレーションヒューズ、PWRTEによってパワーアップタイマをイネーブル(=1の時)またはディスエーブル(=0の時)にすることができます(4.6章)。

パワーアップタイムの遅延時間はチップによっても、VDDと温度によってもかなり差があります。詳細にはDC特性をご覧下さい。

<u>オシレータスタートアップタイマ(OST)</u>: オシレータスタートアップタイマ(OST)ではPWRTの遅延時間が終了した後1024オシレータサイクルの遅延時間(OSC1入力から)をつくります。これによってクリスタルオシレータまたはセラミック共振が始まって安定するのを保証できます。

OSTのタイムアウトはXT、PL、HSに対して、パワーオンリセット時あるいはSLEEPモードから再動作時のみ有効になります。

<u>タイムアウトシーケンス</u>: パワーアップ時のタイムアウトシケーンスはまず、PORが終了した後PWRTのタイムアウトが有効となります。それから、TOSTがアクティブになります。全体のタイムアウトはオシレータの構成とPWRTEヒューズの状態により差があります。例えば、PWRTが0 (PWRTがディスエーブル) にセットされているときの RC モードでは、まったくタイムアウトが起こりません。表4.2.1に、パワーアップ時とSLEEPからのウェークアップ時のタイムアウトを示します。

PORからタイムアウトが起こるので、$\overline{MCLR}$が十分長くローに保持された場合、タイムアウトは無効になります。それから、$\overline{MCLR}$をハイにするとただちに実行が始まります。これによって結合された複数のPIC16C84の動作を同期させるためやテストに便利です。

表4.2.1　各状況下でのタイムアウト

オシレータ構成	パワーアップ		SLEEPからのウェークアップ
	PWRTE = 1	PWRTE = 0	
XT, HS, LP	72 ms + 1024 tosc	1024 tosc	1024 tosc
RC	72 ms	-	-

④

 Preliminary

3.10 数値演算と論理演算ユニット(ALU)

ALUは8ビット幅で加算、減算、シフト、論理動作ができます。特に説明しないかぎり、通常演算動作は通常2の補数を扱います。2オペランド命令では、普通1つのオペランドがワーキングレジスタ(Wレジスタ)かアキュムレータで、他方のオペランドはファイルレジスタか即値データです。シングルオペランド命令ではそのオペランドはWレジスタかファイルレジスタのどちらかです。

3.11 Wレジスタ

Wレジスタは ALU動作で使われる8ビットのワーキングレジスタ(またはアキュムレータ)で、このレジスタはデータメモリの中に置かれていません。

3.12 割込み

PIC16C84には4個の割込み要因があります。

- PB0/INTピンからの外部割込み
- TMR0タイマ カウンタオーバーフロー時の割込み
- データEEPROMライト終了時の割込み
- RB<7:4>ピン信号変化時の割込み

割込みコントロールレジスタ(INTCON、アドレス0Bh)はそれぞれの割込み要求をフラグビットに記録します。それには個々と全体の割込みイネーブルビットもあります。

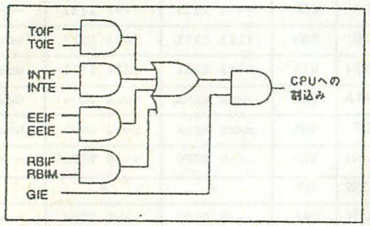
全体の割込みイネーブルビット、GIE(INTCON<7>)は(セットされている時)すべてのマスクされていない割込みをイネーブルにし、または(クリアされた時)すべての割込みをディスエーブルにします。それぞれの割込みをINTCONレジスタの対応するイネーブルビットによってディスエーブルにできます。GIEはリセットによってクリアされます。

RETFIE命令を使うと、割込みから復帰すると同時に割込みをイネーブルに設定することができます。

INTピン割込み、RBポート変化時割込み、TMR0オーバーフロー時割込みのフラグは、INTCONレジスタに含まれています。

割込みを受け付けたとき、それ以降の割込みをディスエーブルにするためにGIEがクリアされ、リターンアドレスがスタックにプッシュされ、PCに0004hがロードされます。いったん割込みサービスルーチンに入ると、割込みフラグビットをポーリングすることによって割込み要因を調べることができます。次の割込みを受け付けるために再び割込みをイネーブルする前にソフトウエアで割込みフラグビットをクリアする必要があります。

図3.12.1 割込み回路



注意1: 個々の割込みフラグビットは、これに対応するマスクビットまたはGIEビットのステータスに関係なくセットされます。

注意2: グローバル割込みイネーブル(GIE)ビットがクリア中に割込みが発生すると、割込みサービスルーチン(RETFIE命令)が誤ってGIEビットを再セットしてしまうことがあります。図3.12.2を参照してください。この誤動作が発生するのは次のケースです。

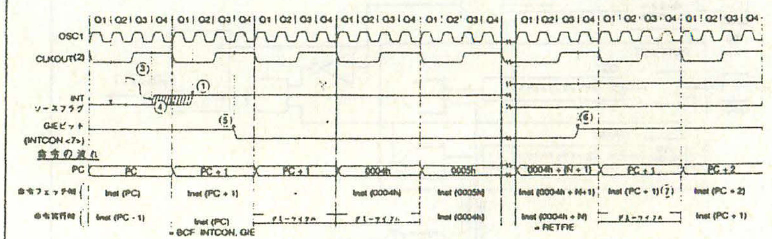
1. 割込みに肯定応答が返されている間に命令がGIEビットをクリアした場合。
2. プログラムが割込みベクタを分岐し、割込みサービスルーチンを実行した場合。
3. 割込みサービスルーチンがRETFIE命令の実行を終了した場合。この結果GIEビットがセットされ(割込みイネーブル)、プログラムは割込みをディスエーブルした命令のすぐ後の命令に戻ります。

全体的に確実に割込みをディスエーブルしたいときは、

1. 次に示すように、命令を実行して確実にGIEビットをクリアしてください。

```
LOOP BCF    INTCON, GIE  ; Disable Global
                         ;  Interrupts
     BTFSC  INTCON, GIE  ; Global Interrupts
                         ;  Disabled?
     GOTO   LOOP         ; NO, try again
                         ; Yes, continue with
                         ;  program flow
```

図3.12.2 GIEのクリアと割込みの発行



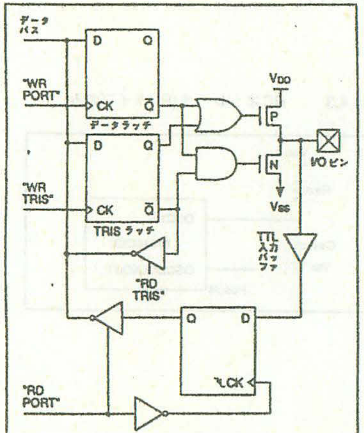
注意:
1. INTFフラグはここで(Q1ごとに)サンプルされます。
2. CLKOUTが可能なのはRCオシレータ・モードだけです。
3. Q4-Q1サイクル中いつでもINTソースフラグをイネーブルにセットできます。
4. GIEビットは命令bcf INTCON GIEのQ4でクリアされます。
5. GIEビットは命令RETFIEのQ4サイクルでセットされます。
6. RERFIE命令が終了したためGIEビットがイネーブルにセットされます。すなわちbcf INTCON GIEはオーバーライトされます。
7. INFソースフラグはRETFIE命令を実行する前にクリアしておかなければなりません。
このフラグがセットされたままだと、プロセッサはこの割込みを引続き実行します。

③

5.0 周辺回路の概要

PIC16C84にはPORTA（5ビット）とPORTB（8ビット）の2個のI/Oポートで構成された13個のI/Oピンがあります。また、PIC16C84には8ビット幅の汎用タイマ／カウンタTMR0が1個付いています。このTMR0は8ビットのプログラマブルプリスケーラをもち、ウオッチドッグタイマから離れた位置に独立して配置されています。この他、PIC16C84には64x8のEEPROMデータメモリが付いており、このデータメモリへは8ビットのデータレジスタとアドレスレジスタからアクセスできるようになっています。

図5.1.1　RA0—RA3ピンのブロック図

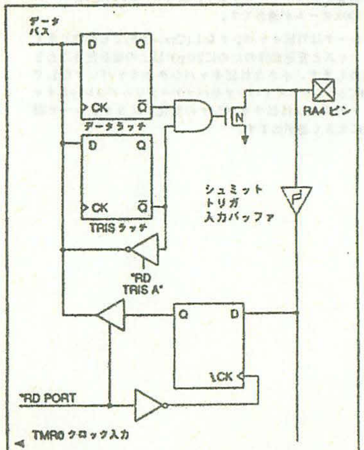


注意1：I/OピンにはVDDおよびVssへのプロテクションダイオードが付いています。

5.1 PORTA

PORTAはピンRA<4:0>をもった5ビット幅のポートです。ポートピンRA<3:0>は双方向、RA4はオープンコレクタ出力をもったポートです。ポートAはファイルレジスタ05hです。対応した方向のコントロールレジスタTRISAはアドレス85hのレジスタファイルのページ1に配置されています。TRISAは5ビット幅のレジスタで、ビット<4:0>が物理的に実現されています。図5.1.1と5.1.2に、PORTAのピンのブロック図を示します。

図 5.1.2　RA4ピンのブロック図



注意1：I/OピンにはVssへのプロテクションダイオードが付いています。

500KHz未満で動作するクリスタルオシレータ構成では、PORTA<0>(RA0)が切り替わるときに、デバイスが疑似内部Qクロックを生成することがあります。RCモードの外部クロックではこのような疑似クロックは発生しません。これを避けるためには、RA0ピンをスタティックに保たなければなりません。したがって、入力／出力モードでRA0をトグルしてはいけません。

表5.1.1　PORTAの機能

ポートピン	ビット	ピンの機能	別の機能
RA0	bit0	入力／出力ポート。TTL入力レベル	-
RA1	bit1	入力／出力ポート。TTL入力レベル	-
RA2	bit2	入力／出力ポート。TTL入力レベル	-
RA3	bit3	入力／出力ポート。TTL入力レベル	-
RA4/T0CKI	bit4	入力／出力ポート。出力はオープンコレクタタイプ。入力はシュミットトリガタイプ。	TMR0タイマ／カウンタ用の外部クロック入力

表5.1.2 PORTAレジスタの概要

レジスタの名称	機能	アドレス	パワーオンリセット値
PORTA	リード時PORTAピン ライト時PORTAラッチ	05h	---x xxxx
TRISA	PORTAのデータ方向レジスタ	85h	---1 1111

注：　1: x=不定、-=影響なし、0をリード
　　2: ほかのリセット状態におけるレジスタのリセット値には表5.1.1を参照

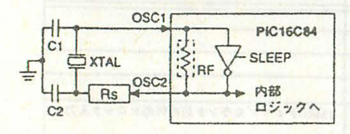
⑥

4.4 オシレータの設定

4.4.1 オシレータのタイプ

PIC16C84は4種類のオシレータオプションで動作することができます。2個のコンフィギュレーションヒューズ(FOSC1とFOSC0)をプログラムして、この4種類のモードを選択することができます。

図4.4.1 クリスタル動作
(またはセラミック共振)
(HS、XT、LPオシレータの設定)



-Rsは、Hsモード、または、ATストリップカットクリスタルのXTモードで必要となる可能性があります。C1とC2の推奨値は表4.4.1と表4.4.2をご覧ください。

表4.4.1 セラミック共振のキャパシタの選択

オレシータのタイプ	共振周波数	キャパシタレンジ C1 = C2
XT	455 KHz	150 - 330 pF
	2.0 MHz	20 - 330 pF
	4.0 MHz	20 - 330 pF
HS	10.0 MHz	20 - 200 pF

注意：より大きいキャパシタほどオシレータの安定性を増しますがスタートアップタイムも増えます。これらの値は設計の目安のためだけです。各共振素子には独自の特性がありますので、外部部品の適当な値については製造メーカに問い合わせてください。

4.4.2 クリスタルオシレータ

XT、HS、LPモードでは発進を実現するためにOSC1とOSC2ピンにクリスタルまたはセラミック素子を接続します。(図4.4.1参照)。

図4.4.2 外部クロック入力動作
(HS、XT、LPオシレータ設定)

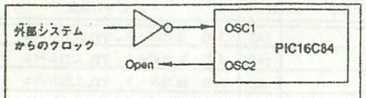


表4.4.2 クリスタルオシレータのキャパシタの選択

Osc タイプ	周波数	C1	C2
LP	32 KHz	68 - 100 pF	68 - 100 pF
	200 KHz	15 - 30 pF	15 - 30 pF
XT	100 KHz	68 - 150 pF	150 - 200 pF
	2 MHz	15 - 30 pF	15 - 30 pF
HS	4 MHz	15 - 30 pF	15 - 30 pF
	10 MHz	15 - 47 pF	15 - 47 pF

注意：より大きいキャパシタほどオシレータの安定性を増しますがスタートアップタイムも増えます。これらの値は設計の目安のためだけです。Rsは低駆動レベル特性をもったクリスタルをオーバードライブしないようにXTモードと同様にHSモードに必要かもしれません。各共振素子には独自の特性がありますので、外部部品の適当な値については製造メーカに問い合わせてください。

4.4.3 RCオシレータ

タイミング精度のいらないアプリケーションではRCデバイスオプションでコストを節約できます。RCオシレータ周波数は電源電圧、抵抗(Rext)、キャパシタ(Cext)の値、動作温度の関数です。さらに、オシレータ周波数は通常の製造パラメータによるチップのばらつきによっても差があります。その上、パッケージタイプによるリードフレームの違いによってもオシレーション周波数に、特に、小さいCext値で、影響があります。使用している外付けのRとCの誤差によるばらつきをも考慮にいれる必要があります。図4.4.3にR/CのどのようなPIC16C84に接続するかを示しています。2.2Kオーム以下のRext値では、オシレータ動作が不安定になるか、完全に停止してしまいます。非常に高いRext値(1Mオームなど)では、オシレータがノイズ、湿度、漏れ電流の影響を受けやすくなります。従って、Rextは5Kオームから100Kオームが適当です。

オシレータは外部キャパシタなし(Cext=0pF)でも動作しますが、ノイズと安定動作のために20pF以上の値を使うことをお勧めします。小さな外部キャパシタかキャパシタなしでは、PCBトレースキャパシタやパッケージリードフレームキャパシタのような外部キャパシタの変化によりオシレータ周波数に大きく差が出ます。

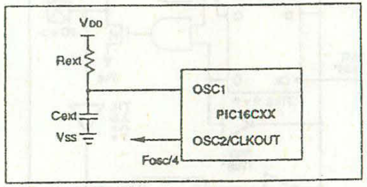
通常の製造チップでのRC周波数のばらつきについては11.0章をご覧下さい。大きなR(漏れ電流のばらつきが大きなRでRC周波数に影響を与えるため)と小さなC(入力キャパシタのばらつきがRC周波数に影響を与えるため)でばらつきが大きくなります。

与えられたR、C、VDD値での動作温度における周波数のばらつきと同様に与えられたRext/Cext値での周波数のばらつきに関して11.0章の特性をご覧下さい。

RCモードでは4分割したオシレータ周波数がOSC2/CLKOUTピンに出力されます。この4分周波数は、テストや他のロジックとの同期に使用できます（タイミングについては図3.2.1を参照してください）。

デバイスがSLEEPからウェークアップすると、ウェークアップの原因となったイベントが何であっても、常にWDTがクリアされます。

図4.4.3 RCオシレータ(RCタイプのみ)



⑤

5.4.4 プリスケーラ設定の切り替え

プリスケーラの設定はすべてソフトウエアコントロールのもとで行なわれます。例えば、プログラム実行中にそれをクリアすることができます。プリスケーラの設定をTMR0からWDTに変更するには、不必要にデバイスをRESETすることを避けるために、次のような命令シーケンス（例5-2参照）を実行する必要があります。

例5-2：プリスケーラ設定の切り替え
(TMR0→WDT)

```
1. BCF      STATUS, RP0   ; バンク0
2. CLRF     TMR0          ; TMR0をクリア
3. BSF      STATUS, RP0   ; バンク1
4. CLRWDT                 ; WDTとプリスケーラを
                          ; クリア
5. MOVLW    B'xxxx1xxx'   ; 新しいプリスケール
6. MOVWF    OPTION        ; 値を選択する。
7. BCF      STATUS, RP0   ; バンク0
```

プリスケーラをWDTからTMR0モジュールに変更したいときは、例5-3のシーケンスを実行してください。WDTがディスエーブルに設定されていても、必ずこの予防措置を実行する必要があります。

例5-3：プリスケーラ設定の切り替え
(WDT→TMR0)

```
1. CLRWDT                 ; WDTとプリスケーラを
                          ; クリア
2. MOVLW    B'xxxx0xxx'   ; TMR0、新しい
                          ; プリスケール値、
                          ; クロックソースを
                          ; 選択。
3. MOVWF    OPTION        ;
```

表5.4.1 TMR0レジスタの概要

レジスタ名称	機能	アドレス	パワーオンリセット値
TMR0	タイマ／カウンタレジスタ	01h	XXXX XXXX
OPTION	TMR0のコンフィギュレーションとプリスケーラの設定ビット。図5.5.1参照	81h	1111 1111
INTCON	TMR0オーバーフロー割込みフラッグとマスクビット。図3.12.3参照	0Bh	0000 000X

表5.4.2 TMR0と連結するレジスタ

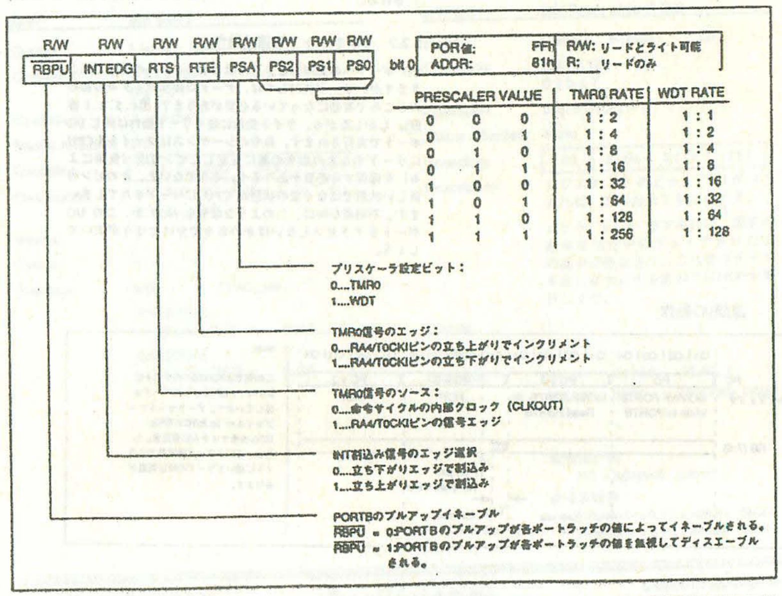
アドレス	名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
01	TMR0	TIMER0							
0B/8B	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
81	OPTION	RBPU	INTEDG	RTS	RTE	PSA	PS2	PS1	PS0
85	TRISA	—	—	—	TRISA 4	TRISA 3	TRISA 2	TRISA 1	TRISA 0

— ＝実行されない位置。0としてリードします。
▒＝TMR0は、使用しません。

5.5 OPTION レジスタ

OPTIONレジスタ（アドレス81h）はプリスケーラ、外部INT割込みエッジ選択、TMR0、PORTBの小さいプルアップを設定するためのいろいろなコントロールビットを含んでいるレジスタで、リード、ライト可能です。

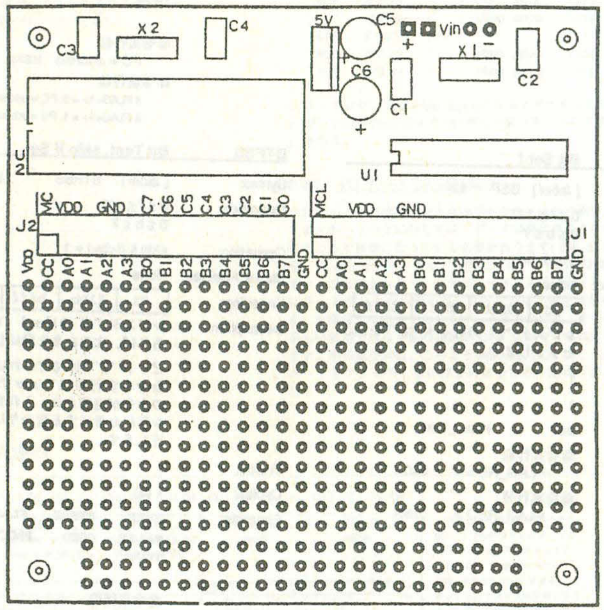
図5.5.1 OPTIONレジスタ



⑧

ＰＩＣシリーズ用プリント基板

☆ワンチップＲＩＳＣライクプロセッサＰＩＣシリーズ専用基板です
☆ＰＩＣチップ、電源、Ｉ／Ｏコネクタ周りは全て専用のプリントパターンとなっています
☆２９×１３グリットのユニバーサルエリアを設けてあるのでマイコン制御機器の開発・製作に最適です
☆ＰＩＣ１６Ｃ５４，５６，７１，８４　（１８ピン）回路とＰＩＣ１６Ｃ５５，５７　（２８ピン）回路とを同時に使用することができます。



技術資料付　１枚　￥５００．

PIC16C84使用 AKI-PICプラチナキット

☆ＰＩＣ１６Ｃ８４を使用したワンボードマイコンキットです
☆Ｘ'ｔａｌ，ＣＲ等、周辺回路パーツ付属
☆プログラム書込に関する資料とライター用のパーツ、また、アセンブラ、シュミレータ等のソフトが付属

技術資料
専用基板　一式付属　￥３，０００．

(有)秋月電子

SUBWF — Subtract W from f

Syntax:	[*label*] SUBWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f)-(W) → (dest)
Status Affected:	C, DC, Z
Encoding:	00 | 0010 | dfff | ffff
Description:	レジスタ "f" からW レジスタの内容を（2の補数を使って）引きます。この結果を、d=0であればWレジスタに、d=1であればレジスタ "f" に書き込みます。
Words:	1
Cycles:	1

Example 1:

```
SUBWF  REG1,1
命令実行前
    REG1 = 0
    W = 1
    C = ?
命令実行後
    REG1 = FF
    W = 1
    C = 0   ；結果は負
```

Example 2:

```
命令実行前
    REG1 = FF
    W = 0
    C = ?
命令実行後
    REG1 = FF
    W = 0
    C = 1   ；結果は正
```

SWAPF — Swap f

Syntax:	[*label*] SWAPF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	f<0:3> → d<4:7>, f<4:7> → d<0:3>;
Status Affected:	None
Encoding:	00 | 1110 | dfff | ffff
Description:	レジスタ "f" の上位ビットと下位ビットを入れ替えます。この結果を、d=0であればWレジスタに、d=1であればレジスタ "f" に書き込みます。
Words:	1
Cycles:	1

Example:

```
SWAPF  REG, 0
命令実行前
    REG = 0xA5
命令実行後
    REG = 0xA5
    W   = 0x5A
```

TRIS — Load TRIS Register

Syntax:	[*label*] TRIS f
Operands:	5 ≤ f ≤ 7
Operation:	W → TRIS register f;
Status Affected:	None
Encoding:	00 | 0000 | 0110 | 0fff
Description:	これは、PIC16C5X製品とのコード互換性を保つために用意された命令です。TRISレジスタのリードとライトの両方が可能で、ユーザはこのレジスタを直接アドレス指定できます。
Words:	1
Cycles:	1
Example:	

将来のPIC16CXX製品との上位互換性を維持するため、この命令は使わないでください。

XORLW — Exclusive OR literal with W

Syntax:	[*label*] XORLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) .XOR. k → (W)
Status Affected:	Z
Encoding:	11 | 1010 | kkkk | kkkk
Description:	W レジスタの内容と8ビットのリテラル "k" とのXORを取り、この結果をWレジスタに書き込みます。
Words:	1
Cycles:	1

Example:

```
XORLW  0xAF
命令実行前
    W = 0xB5
命令実行後
    W = 0x1A
```


XORWF — Exclusive OR W with f

Syntax:	[*label*] XORWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) .XOR. (f) → (dest)
Status Affected	Z
Encoding:	00 | 0110 | dfff | ffff
Description:	W レジスタの内容とレジスタ "f" のXORを取り、この結果を、d=0であればW レジスタに、d=1であればレジスタ "f" に書き戻します。
Words:	1
Cycles:	1

Example:

```
XORWF  REG, 1
命令実行前
    REG = 0xAF
    W   = 0xB5
命令実行後
    REG = 0x1A
    W   = 0xB5
```

⑫

INCF — Increment f

Syntax:	[*label*] INCF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) + 1 → (dest)
Status Affected:	Z
Encoding:	00 | 1010 | dfff | ffff
Description:	レジスタ "f" を減分し、この結果を、d=0であればWレジスタに、d=1であればレジスタ "f" に書き戻します。
Words:	1
Cycles:	1
Example:	INCF CNT, 1 命令実行前 CNT = 0xFF Z = 0 命令実行後 CNT = 0x00 Z = 1

INCFSZ — Increment f, skip if 0

Syntax:	[*label*] INCFSZ f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) + 1 → (dest), skip if result = 0
Status Affected:	None
Encoding:	00 | 1111 | dfff | ffff
Description:	レジスタ "f" の内容を増分し、この結果を、d = 0 であればW レジスタに、d=1であればレジスタ "f" に書き戻します。 結果が 0の場合は、既にフェッチされている次の命令を廃棄し、この代わりにNOPを実行します。INCFSZ は2サイクルの命令です。
Words:	1
Cycles:	1 (2)
Example:	HERE INCFSZ CNT, 1 GOTO LOOP CONTINUE • • • 命令実行前 PC = address HERE 命令実行後 CNT = CNT + 1 if CNT = 0, PC = address CONTINUE if CNT ≠ 0, PC = address HERE + 1

IORLW — Inclusive OR Literal with W

Syntax:	[*label*] IORLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) .OR. (k) → (W)
Status Affected:	Z
Encoding:	11 | 1000 | kkkk | kkkk
Description:	W レジスタの内容と8 ビットのリテラル "k" のORを取り、この結果をWレジスタに書き戻します。
Words:	1
Cycles:	1
Example:	IORLW 0x35 命令実行前 W = 0x9A 命令実行後 W = 0xBF

IORWF — Inclusive OR W with f

Syntax:	[*label*] IORWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) .OR. (f) → (dest)
Status Affected:	Z
Encoding:	00 | 0100 | dfff | ffff
Description:	W レジスタとレジスタ "f" の包含的ORを取り、この結果を、d=0であれば Wレジスタに、d=1であればレジスタ "f" に書き戻します。
Words:	1
Cycles:	1
Example:	IORWF RESULT, 0 Before Instruction RESULT = 0x13 W = 0x91 After Instruction RESULT = 0x13 W = 0x93

MOVLW — Move Literal to W

Syntax:	[*label*] MOVLW k
Operands:	0 ≤ k ≤ 255
Operation:	k → (W)
Status Affected:	None
Encoding:	11 | 00XX | kkkk | kkkk
Description:	8ビットのリテラル "k" をWレジスタにロードします。
Words:	1
Cycles:	1
Example:	MOVLW 0x5A 命令実行後 W = 0x5A

MOVF — Move f

Syntax:	[*label*] MOVF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) → (dest)
Status Affected:	Z
Encoding:	00 | 1000 | dfff | ffff
Description:	レジスタ "f" の内容を宛先 "d" に移動します。d=0であれば宛先はWレジスタです。d=1であれば宛先はファイルレジスタ "f" 自体ですが、このときステータスフラグZが変化するので、ファイルレジスタのテストに便利です。
Words:	1
Cycles:	1
Example:	MOVF FSR, 0 命令実行後 W = value in FSR register

MOVWF — Move W to f

Syntax:	[*label*] MOVWF f
Operands:	0 ≤ f ≤ 127
Operation:	(W) → (f)
Status Affected:	None
Encoding:	00 | 0000 | 1fff | ffff
Description:	Wレジスタからレジスタ "f" にデータを移動します。
Words:	1
Cycles:	1
Example:	MOVWF OPTION 命令実行前 OPTION = 0xFF W = 0x4F 命令実行後 OPTION = 0x4F W = 0x4F

NOP — No Operation

Syntax:	[*label*] NOP
Operands:	None
Operation:	No operation
Status Affected:	None
Encoding:	00 | 0000 | 0XX0 | 0000
Description:	何の操作も行われません。
Words:	1
Cycles:	1
Example:	NOP

⑪

10.0 タイミング・ダイアグラム

図10.0.1　TOCKIタイミング

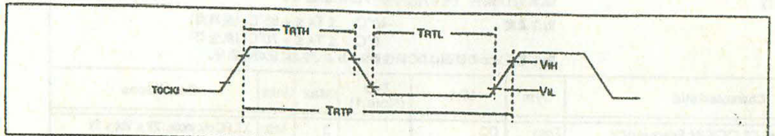
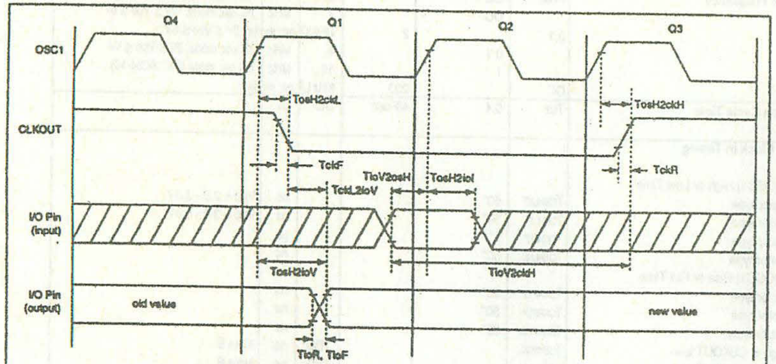


図10.0.2　I/Oポートの入力／出力タイミング



Note 1:　すべてのタイミングは、規定の容量性負荷を課してテストしてあります。
Note 2:　CLKOUTが可能なのはPCオシレータモードだけです。

　Preliminary　

索引

商　標

⑭

　Preliminary　

9.3 DC特性： PIC16C84-04（民生用、産業用）
PIC16C84-10（民生用、産業用）
PIC16LC84-04（民生用、産業用）

DC特性

標準動作条件（その他の条件は記載通り）
動作温度　-40°C ≤ TA ≤ +85°C（産業用）
0°C ≤ TA ≤ +70°C（民生用）
動作電圧VDDの範囲はDC特性表9.1および9.2に記載の通り。

Characteristic	Sym	Min	Typ (Note 1)	Max	Units	Conditions
Input Low Voltage						
I/O ports	VIL	Vss		0.2 VDD	V	
RB<7:4>		Vss		0.2 Vss	V	2.0V ≤ VDD ≤ 3.0V
		Vss		0.6 Vss	V	VDD ≥ 3.0V
MCLR, TOCKI		Vss		0.2 VDD	V	
OSC1 (in RC configuration)		Vss		0.1 VDD		Note 2
OSC1 (in XT, HS and LP configuration)		Vss		0.3 VDD	V	
Input High Voltage						
I/O ports	VIH	2.0		VDD	V	VDD ≤ 5.5V (note 5)
		0.7 VDD				For entire VDD range (note 5)
MCLR, TOCKI, OSC1 (in RC configuration)		0.85 VDD		VDD	V	Note 2
OSC1 (XT, HS and LP configuration)		0.7 VDD		VDD	V	
Input Leakage Current (Notes 3, 4)						
I/O ports RA, RB	IIL			±1	μA	Vss ≤ VPIN ≤ VDD, Pin at hi-impedance
MCLR, TOCKI				±5	μA	Vss ≤ VPIN ≤ VDD
OSC1				±5	μA	Vss ≤ VPIN ≤ VDD , XT, HS and LP osc configuration
PortB Weak Pull-up Current	IPU	50*	250*	400*	μA	VPIN = Vss, VDD = 5.0V
Output Low Voltage						
I/O Ports	VOL			0.6	V	IOL = 8.5mA, VDD = 4.5V, -40°C to +85°C
				0.6	V	IOL = 6.0mA, VDD = 4.5V, -40°C to +125°C
OSC2/CLKOUT				0.6	V	IOL = 1.6mA, VDD = 4.5V, -40°C to +85°C
(RC osc configuration)				0.6	V	IOL = 1.2mA, VDD = 4.5V, -40°C to +125°C
Output High Voltage						
I/O Ports (Note 4)	VOH	VDD-0.7			V	IOH = -3.0mA, VDD = 4.5V, -40°C to +85°C
		VDD-0.7			V	IOH = -2.5mA, VDD = 4.5V, -40°C to +125°C
OSC2/CLKOUT		VDD-0.7			V	IOH = -1.3mA, VDD = 4.5V, -40°C to +85°C
(RC osc configuration)		VDD-0.7			V	IOH = -1.0mA, VDD = 4.5V, -40°C to +125°C
Data EEPROM						
Endurance	Ed	100,000*	1,000,000		E/W cycles	
VDD for read/write	Vdrw		Full VDD range			
Erase/write cycle time	Tdew		10		ms	
Program EEPROM Memory						See section 11 for more details on programming
Endurance	Ep	100*	1000		E/W cycles	
Erase/write cycle time	Tpew		10		ms	
VDD for read	Vpr		Full VDD range			
VDD for erase/write	Vpw	4.5		5.5	V	

前ページ参照

9.4 AC特性： PIC16C84-04（民生用、産業用）
PIC16C84-10（民生用、産業用）
PIC16LC84-04（民生用、産業用）

AC特性

標準動作条件（その他の条件は記載通り）
動作温度　-40°C ≤ TA ≤ +85°C（産業用）
0°C ≤ TA ≤ +70°C（民生用）
動作電圧VDDの範囲はDC特性表9.1および9.2に記載の通り。

Characteristic	Sym	Min	Typ (Note 1)	Max	Units	Conditions
External CLOCKIN Frequency (Note 2)	Fosc	DC		2	MHz	XT, RC osc mode. 2V ≤ VDD ≤ 6V
		DC		4	MHz	XT, RC osc mode. 3V ≤ VDD ≤ 6V
		DC		10	MHz	HS osc mode (PIC16C84-10)
		DC		200	KHz	LP osc mode
Oscillator Frequency (Note 2)	Fosc	DC		2	MHz	RC OSC mode. 2V ≤ VDD ≤ 6V
		DC		4	MHz	RC osc mode. 3V ≤ VDD ≤ 6V
	0.1		2		MHz	XT osc mode. 2V ≤ VDD ≤ 6V
		0.1		4	MHz	XT osc mode. 3V ≤ VDD ≤ 6V
		1		10	MHz	HS osc mode (PIC16C84-10)
	DC		200		KHz	LP osc mode
Instruction Cycle Time (Note 2)	TCY	0.4	4/Fosc	DC	μs	
External Clock in Timing (Note 4)						
Clock in (OSC1) High or Low Time						
XT oscillator type	TCKHLXT	60*			ns	VDD = 2.0 - 3.0V
XT oscillator type	TCKHLXT	50*			ns	VDD = 3.0 - 6.0V
LP oscillator type	TCKHLLP	2*			μs	
HS oscillator type	TCKHLHS	50*			ns	
Clock in (OSC1) Rise or Fall Time						
XT oscillator type	TCKRFXT	25*			ns	
LP oscillator type	TCKRFLP	50*			ns	
HS oscillator type	TCKRFHS	25*			ns	
OSC1 high to CLKOUT low	TOSH2CKL			TBD	ns	Note 6
OSC1 high to CLKOUT high	TOSH2CKH			TBD	ns	Note 6
CLKOUT output rise time	TCKR			TBD	ns	Note 6
CLKOUT output fall time	TCKF			TBD	ns	Note 6
RESET Timing						
MCLR Pulse Width (low)	TMCL	350*			ns	VDD = 2.0 - 3.0V
MCLR Pulse Width (low)	TMCL	150*			ns	VDD = 3.0 - 6.0V
TMR0 Input Timing, No Prescaler						
TMR0 High Pulse Width	TRTH	0.5 TCY + 20*			ns	Note 3
TMR0 Low Pulse Width	TRTL	0.5 TCY + 20*			ns	Note 3
TOCKI Input Timing, With Prescaler						
TOCKI High Pulse Width	TRTH	50*			ns	VDD = 2.0 - 3.0V
TOCKI High Pulse Width	TRTH	30*			ns	VDD = 3.0 - 6.0V
TOCKI Low Pulse Width	TRTL	50*			ns	VDD = 2.0 - 3.0V
TOCKI Low Pulse Width	TRTL	20*			ns	VDD = 3.0 - 6.0V
TOCKI Period	TRTP	TCY + 40 * / N			ns	Note 3. Where N = prescale value (2,4, ..., 256)
Watchdog Timer Timeout Period (No Prescaler)	TWDT	7*	18*	33*	ms	VDD = 5V, -40°C to +125°C
OST/PWRT Timings						
Oscillation Start-up Timer Period	TOST		1024 tosc		ms	tosc = OSC1 period
Power-up timer period	TPWRT	28*	72*	32*	ms	VDD = 5V, -40°C to +125°C

＊これらのパラメータは標準的なもので、テストによって検証されたものではありません。

次ページ参照

9.0 電気的特性

絶対最大定格*

バイアス時周囲温度 .... -55 から +125℃
保存温度 .... -65℃から +150℃
Vssに対する各ピンの電圧
（VDDとMCLRを除く） .... -0.6Vから VDD +0.6V
Vssに対するVDDの電圧 .... 0 から +7.5 V
Vssに対するMCLRの電圧
（Note2） .... 0から +14 V
総合消費電力（Note 1） .... 800 mW
Vssピンから流れ出す最大電流 150mA
VDDピンへ流れ込む最大電流 100mA
入力クランプ電流、IIK (VI < 0 または VI > VDD) .... ±20mA
出力クランプ電流、IOK (V0 < 0 または V0 > VDD) .... ±20mA
各I/Oピンの最大出力シンク電流 25mA
各I/Oピンの最大出力ソース電流 20mA
I/O PORTAの最大出力シンク電流 80mA
I/O PORTBの最大出力シンク電流 150mA
I/O PORTAの最大出力ソース電流 50mA
I/O PORTBの最大出力ソース電流 100mA

Notes: 1. 総合消費電力はパッケージに対して800mWを超えてはいけません。消費電力は次のように計算します：

$$P_{dis} = V_{DD} \times (I_{DD} - \Sigma I_{oh}) + \Sigma \{(V_{DD} - V_{oh}) \times I_{oh}\} + \Sigma (V_{ol} \times I_{ol})$$

2. MCLRピンでのVss以下のスパイク電圧は、80mA以上の電流も含めて、ラッチアップを起こすことがあります。したがって、MCLRピンにローレベルを与えるときは、そのピンを直接Vssに引く代わりに50-100Ωの直列抵抗を使う必要があります。

*Notice：“絶対最大定格”に記載されている値を超えたストレスを与えると、デバイスに対して永久的なダメージを与えることがあります。これはストレスに対する定格のみであって、この定格の動作表に示されている値以上の、またはそれ以外の条件における、デバイスの機能動作、ACとDCのパラメータ定格の保証は意図していません。長期間、最大定格の条件においておくことはデバイスの信頼性に影響を与えます。

9.1 DC特性：PIC16C84-04（民生用、産業用）PIC16C84-10（民生用、産業用）

DC特性

標準動作条件（その他の条件は記載通り）
動作温度 -40°C ≤ TA ≤ +85°C（産業用）
0°C ≤ TA ≤ +70°C（民生用）
動作電圧 VDD = 4.0V から 6.0V

Characteristic	Sym	Min	Typ (Note 1)	Max	Units	Conditions
Supply Voltage	VDD	4.0		6.0	V	XT, RC and LP osc configuration
		4.5		5.5	V	HS osc configuration
RAM Data Retention Voltage (Note 2)	VDR		1.5		V	Device in SLEEP mode
VDD start voltage to guarantee Power-on Reset	VPOR		Vss		V	See Section 4.2 for details on power on reset
VDD rise rate to guarantee Power-on Reset	SVDD	0.05*			V/ms	See Section 4.2 for details on power on reset
Supply Current (Note 3)	IDD		7.3	10	mA	Fosc = 4 MHz, VDD = 5.5V during EEPROM programming
			1.8	4.5	mA	Fosc = 4 MHz, VDD = 5.5V (Note 5)
			35	400	μA	Fosc = 32 KHz, VDD = 4.0V, WDT disabled, LP osc configuration
			5	10	mA	Fosc = 10 MHz, VDD = 5.5V, HS osc configuration (PIC16C84-10 only)
Power Down Current (Note 4)	IPD		7	100	μA	VDD = 4.0V, WDT enabled, -40°C to +85°C
		1.0	100		μA	VDD = 4.0V, WDT disabled, 0°C to +70°C
			1.0	100	μA	VDD = 4.0V, WDT disabled, -40°C to +85°C

*これらのパラメータは標準的なもので、テストによって検証されたものではありません。

Note1： “Typical”と表示されている欄のデータは25℃における評価を元にしています。このデータは設計の参考用のみで、Microchip Technologyでテストまたは保証をしていません。

Note2： SLEEPモード時にRAMデータを失うことなくVDDが低下できる限界です。

Note3： 消費電流はおもに動作電圧と周波数に関係します。I/Oピンの負荷、スイッチングスピード、オシレータタイプ、内部コードの実行パターン、温度なども消費電流に影響を与えます。アクティブ動作モードにおけるIDD測定のテスト条件は：
OSC1はVDDとVss範囲内の外部矩形波；すべてのI/Oピンはトライステート、VDDにプルアップ、T0CKI= VDD、MCLR = VDD；WDTは指示されているとおりにイネーブル／ディスエーブルさせます。

Note4： SLEEPモード時のパワーダウン電流はオシレータタイプには無関係です。パワーダウン電流はチップがSLEEPモードで、すべてのI/Oピンがハイインピーダンス状態とし、VDDとVssには電圧がかかっている状態で測定されます。

Note5： RCオシレータの設定では、Rextの電流は含まれません。その抵抗からの電流は次の式から推定できます：
$I_R = V_{DD}/2R_{ext}$ (mA)、Rext（外部プルアップ抵抗）はkオーム。

[redacted]

RETFIE — Return from Interrupt

Syntax: [*label*] RETFIE

Operands: None

Operation: TOS → PC,
1 → GIE;

Status Affected: None

Encoding:

00	0000	0000	1001

Description: 割込みからの復帰。まずスタックをポップアップし、次にスタックのトップ（TOS）をPCにロードします。その後、GIEビットをセットして割込みをイネーブルにしてください。GIEはグローバル割込みイネーブルビットです（INTCON<7>）。RETFIEは2サイクルの命令です。

Words: 1

Cycles: 2

Example:

```
RETFIE
割込み後
    PC  =  TOS
    GIE =  1
```

RETLW — Return Literal to W

Syntax: [*label*] RETLW k

Operands: $0 \le k \le 255$

Operation: k → W; TOS → PC;

Status Affected: None

Encoding:

11	01XX	kkkk	kkkk

Description: まず8ビットのリテラル "k" をWレジスタにロードし、次にプログラムカウンタにスタックのトップ（リターンアドレス）をロードします。これは2サイクルの命令です。

Words: 1

Cycles: 2

Example:

```
      CALL  TABLE ; W contains table offset
                  ; value
      .           ; W now has table value
      .
      .
TABLE ADDWF PC    ; W = offset
      RETLW k1    ; Begin table
      RETLW k2    ;
      .
      .
      .
      RETLW kn    ; End of table
```

命令実行前
W = 0x07

命令実行後
W = value of k7

RETURN — Return from Subroutine

Syntax: [*label*] RETURN

Operands: None

Operation: TOS → PC;

Status Affected: None

Encoding:

00	0000	0000	1000

Description: サブルーチンからの復帰。まずスタックをポップアップし、次にスタックのトップ（TOS）をプログラムカウンタにロードします。これは2サイクルの命令です。

Words: 1

Cycles: 2

Example:

```
RETURN
割込み後
    PC = TOS
```


RLF — Rotate Left f through Carry

Syntax: [*label*] RLF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

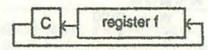
Operation: f<n> → d<n+1>, f<7> → C, C → d<0>;

Status Affected: C

Encoding:

00	1101	dfff	ffff

Description: レジスタ "f" の内容をキャリーフラグを通過して1ビット左に回転します。この結果を、d=0であればWレジスタに、d = 1 であればレジスタ "f" に書き戻します。



Words: 1

Cycles: 1

Example: RLF REG1,0

命令実行前
REG1 = 11100110
C = 0

命令実行後
REG1 = 11100110
W = 11001100
C = 1

RRF — Rotate Right f through Carry

Syntax: [*label*] RRF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: f<n> → d<n-1>,
f<0> → C,
C → d<7>;

Status Affected: C

Encoding:

00	1100	dfff	ffff

Description: レジスタ "f" の内容をキャリーフラグを通過して1ビット右に回転します。この結果を、d=0であればWレジスタに、d = 1 であればレジスタ "f" に書き戻します。



Words: 1

Cycles: 1

Example: RRF REG1,0

命令実行前
REG1 = 11100110
C = 0

命令実行後
REG1 = 11100110
W = 01110011
C = 1

SLEEP

Syntax: [*label*] SLEEP

Operands: None

Operation: 00h → WDT,
0 → WDT prescaler
1 → $\overline{TO}$,
0 → $\overline{PD}$

Status Affected: $\overline{TO}$, $\overline{PD}$

Encoding:

00	0000	0110	0011

Description: パワーダウンステータスビット（$\overline{PD}$）をクリアし、タイムアウトステータスビット（$\overline{TO}$）をセットし、ウオッチドッグタイマとそのプリスケーラをクリアします。

オシレータの停止と共に、プロセッサはSLEEPモードに入ります。詳しくはSLEEPモードの章をご覧ください。

Words: 1

Cycles: 1

Example: SLEEP

SUBLW — Subtract W from Literal

Syntax: [*label*] SUBLW k

Operands: $0 \le k \le 255$

Operation: k - (W) → (W)

Status Affected: C, DC, Z

Encoding:

11	110X	kkkk	kkkk

Description: 8ビットのリテラル "k" からWレジスタの内容を（2 の補数を使って）引きます。この結果をWレジスタに書き込みます。

Words: 1

Cycles: 1

Example 1: SUBLW 0X02

命令実行前
W = 1
C = ?

命令実行後
W = 1
C = 1 ; result is positive

Example 2:

命令実行前
W = 3
C = ?

命令実行後
W = FF
C = 0 ; result is negative

5.6 EEPROMデータメモリ

PIC16C84には64x8のEEPROMデータメモリが内臓されています。このEEPROMデータメモリに対しては、正規の操作中に（全VDD範囲で）リードとライトを実行できます。このメモリはレジスタファイルのスペースの中に直接マップされている訳ではありません。このメモリへは、EEDATA<08h>とEEADR<09h>の2つのレジスタを介してアクセスします。EEDATA<08h>はリード　ライト用の8ビットデータを保持するレジスタ、EEADR<09h>はアクセス先のEEPROMロケーションのアドレスを保持するレジスタです。このメモリの64バイトは0h-63Hのアドレス範囲に置かれています。さらに、2つのコントロールレジスタEECON1<88h>とEECON2<89h>が用意されています。

EEPROMデータメモリではバイト単位のリードとライトが可能です。バイトライトを実行すると、自動的にそのロケーションのデータが消去され、新しいデータが書き込まれます（消去後ライト）。EEPROMデータメモリは高い消去 ライトサイクルか可能な構成になっています。ライト時間は公称10msで、オンチップタイマで制御できます。実際のライト時間は、チップによっても変わりますし、電圧や温度によっても変化します。正確な限界値についてはAC仕様をご覧ください。

デバイスがコードプロテクトされていると、データメモリにリードやライトを実行できるのはCPUだけになります。つまり、デバイスのプログラマはこのメモリにアクセスできなくなります（リード ライトのディスエーブル）。

5.6.1 EECON1 および EECON2レジスタ

EECON1（アドレス88h）はコントロールレジスタで、下位5ビットだけが物理的に実現されています。上位3 ビットは存在せず、"0" として読み出されます。

コントロールビットRDとWRは、それぞれ、リードとライトを開始します。これらのビットはソフトウェアでしかセットできません。これらのビットは、リードまたはライト操作が完了した時点で、ハードウェアによってリセットされます。WRビットをソフトウェアでクリアできなくしたため、ライト操作が完了する前に誤って操作を打ち切ってしまう恐れは全くなくなりました。

WRENビットを1にセットすると、ライト操作の実行が可能になります。パワーアップ時にはWRENビットは0にセットされています。WRERRビットは、正規の操作中に$\overline{MCLR}$リセットまたはWDTタイムアウト・リセットによってライト操作への割込みが行われたときにセットされます。このような場合には、ユーザはリセット後にWRERRビットをチェックし、そのロケーションにリライトを実行することができます。上記のようなリセットによる割込みが発生しても、EEDATAレジスタとEEADRレジスタの中のデータとアドレスは変わりません。

EEIFビットは、ライトが完了した時にセットされる割込みフラグです。このビットはソフトウェアの中でクリアしなければなりません。

EECON2は物理的なレジスタではありません。EECON2にリードを実行しても、"0" が読み出されるだけです。

INDFを使用したEECON2に対して間接アドレッシングアクセスはできません。

図5.6.3.1 EECON1レジスタ

U	U	U	R/W	R/W	R/W	R/W	R/W
-	-	-	EEIF	WRERR	WREN	WR	RD

bit0

アドレス: 88h
リセット値: 0000x000b

R/W: リードおよびライト可能
R: リードのみ

RD: リードコントロールビット。RD=1にセットするとEEPROMのリードが開始されます。リードは1クロック・サイクルで終了し、終了時にハードウェアの中でRDがクリアされます。RDビットはソフトウェアの中でしかセットできません（クリアはできません。）

WR: ライトコントロールビット。WR=1にセットするとライトサイクルが開始されます。ライトが終了すると、ハードウェアの中でWRビットがクリアされます。WRビットはソフトウェアの中でしかセットできません（クリアはできません。）

WREN: EEPROMライトイネーブルビット。WREN=0にセットするとEEPROMへのデータライトが禁止されます。WREN=1にセットすると、EEPROMへのライト操作が可能になります。

WRERR: WRERRエラーフラグは、（SLEEPまたは正規操作中に）$\overline{MCLR}$リセットによって、または、（正規操作中に）WDTリセットによって、ライト操作が途中で打ち切られたときにセットされます。

EEIF: EEPROMライト完了時割込みフラグビット。ライト完了時にセットされます。このフラグビットはソフトウェアの中でリセットしなければなりません。これに対応するイネーブルビットはINTCONレジスタのEEIEビットです。

5.6.2 EEPROMデータメモリの読取り

データメモリのロケーションを読み取りたいときは、EEADRレジスタにそのロケーションのアドレスを書き込み、コントロールビットRD（EECON1<0>）をセットしてください。このデータは次のサイクルでEEDATAレジスタに取り込まれますから、ユーザは次の命令でこのデータを読み出すことができます。EEDATAレジスタは、次のリードが実行されるまで、または、ユーザが（ライト操作中に）EEDATAに新しいデータを書き込むまで、この値を保持します。

5.6.3 EEPROMデータメモリへの書込み

EEPROMデータメモリのロケーションにデータを書き込みたいときは、EEADRレジスタにそのロケーションのアドレスを書き込み、EEDATAレジスタにデータを書き込んでください。その後、次のシーケンスを実行してライトを開始してください。

```
movlw   55h
movwf   EECON2
movlw   AAh
movwf   EECON2
bsf     EECON1,WR    ;WRビットをセット
                     ;ライト開始
```

必ず上記のシーケンスで命令を実行しないと（まず55hをEECON2に書き込み、次にAAhをEECON2に書き込み、最後にWRビットをセットしてください）、ライトは開始されません。また、このコードセグメント実行中は割込みをディスエーブルに設定しておかなければなりません。

さらに、EECON1のWRENビットはライトイネーブルの状態にセットしておかなければなりません。WRENビットは、うっかり予期しないコードを実行してデータEEPROMに間違ったデータを書き込んでしまうのを防ぐために用意されたビットです。EEPROMを更新するとき以外は、常にWRENビットをオフに設定しておかれるようお薦めします。さらに、このコードセグメント（WRENビットをイネーブルにセットしてライトを開始するコードセグメント）は、ソフトウェアの誤動作によって誤って実行されるのを防ぐために、他のコードセグメントから離れた位置に置いてください。

ライトが完了すると、ハードウェアの中でWRビットがクリアされ、EEライト完了割込みフラグがセットされます（ビットEEIF）。ユーザはこの割込みをイネーブルすることもできますし、このビットをポーリングすることもできます。EEIFビットはソフトウェアの中でクリアしなければなりません。

注意：データEEPROMメモリのE/Wサイクル時間は、10msの仕様（標準値）を越えることがあります。ライトサイクルが完了したかどうか確かめたいときは、EE割込みを実行するか、または、WRビット（EECON<1>）をポールしてください。どちらの方法でもライトサイクルの完了を確認することができます。

表5.6.2 EEPROMレジスタの概要

レジスタ名	機能	アドレス	パワーオン-リセット値
EEDATA	EEPROM データレジスタ	08h	XXXX XXXX
EEADR	EEPROM アドレスレジスタ	09h	XXXX XXXX
EECON1	EEPROM コントロールレジスタ1	88h	0000 X000
EECON2	EEPROM コントロールレジスタ2	89h	-

5.6.4 疑似書込み保護

PIC16C84には、EEPROMに誤ってライトが実行されるのを防ぐための様々なメカニズムが用意されています。たとえば、パワーアップ時にはWRENビットがクリアされますし、パワーアプタイマ（72msのタイマ）がEEPROMへのライトを防ぎます。

また、ライトの開始シーケンスとWRENビットの組み合わせは、電圧低下、一時的な停電、ソフトウェアの誤動作などによる不正なライトを防ぐのに役立ちます。

5.6.5 EEADRの初期設定

EEADRレジスタには、64バイトまでのデータEEPROMアドレスを指定することができます。このアドレス指定には、EEADRレジスタの8ビットのうち6ビットだけ（EEADR<5:0>）を使用します。このレジスタのビットはパワーアップ時には初期設定されません。

EEADR<7:6>ビットをクリアすると、デバイスの最大IDDが下がります。仕様の記載値は400μsです。EEADR<7:6>がクリアされているときの最大電流は約150μAです。

7.0 命令セットの概要

PIC16CXXの命令はすべて14ビットワードで、命令の種類を指定するOPCODEの部分と、命令の実行方法を詳細に指定する1個または2個以上のオペランドの部分から成り立っています。表7-2には、PIC16CXXの命令セットの概要を、バイト対応、ビット対応、リテラルおよびコントロール操作に分けてリストしてあります。表7-1には、OPCODEフィールドの説明をまとめました。

バイト対応の命令では、"f" をファイルレジスタの指名子、"d" を宛先の指名子として使います。ファイルレジスタ指名子では、命令で使用するファイルレジスタを指定します。

宛先指定子では、命令の実行結果を格納する場所を指定します。"d" がゼロの場合、結果はWレジスタに格納されます。"d" が1の場合、結果は命令で指定されたファイルレジスタに格納されます。

ビット対応の命令では、ビットフィールド指名子 "b" を使って、この命令実行によって影響を受けるビットの番号を選択します。また、ファイル番号指定子 "f" を使って、そのビットが置かれているファイルの番号を指定します。

リテラルおよびコントロール命令では、"k" を使って8ビットまたは11ビットの定数やリテラル値を指定します。

表7.1　　OPCODEフィールドの説明

フィールド	説　明
f	レジスタファイルのアドレス (0x00～0x7F)
W	ワーキングレジスタ (アキュームレータ)
b	8ビットのファイルレジスタの中のビットのアドレス
k	リテラルフィールド、定数データ、または、ラベル
x	無効ロケーション (=0または1) アセンブラはx=0を伴うコードを生成。これはあらゆるソフトウェアツールとの互換性を確保するために推奨されている形式です。
d	宛先指定子：d=0ならば結果をWに格納、 d=1ならばファイルレジスタに格納。 デフォルトはd=1
label	ラベル名
TOS	最上位スタック
PC	プログラムカウンタ
PCLATH	プログラムカウンタのハイラッチ
GIE	グローバル割込みイネーブルビット
WDT	ウォッチドッグタイマ／カウンタ
$\overline{TO}$	タイムアウトビット
$\overline{PD}$	パワーダウンビット
dest	宛先 (Wレジスタまたは指定レジスタファイルのロケーション)
[ ]	オプション
( )	内容
→	割当て先
< >	レジスタビットフィールド
∈	セットを表わす。
イタリック	ユーザ定義用語

命令セットは非常に直交的で、次の3つの基本カテゴリーに分類されます。

- バイト対応の命令
- ビット対応の命令
- リテラルおよびコントロール命令

すべての命令は基本的には1つのシングル命令サイクルの中で実行されますが、命令を実行した結果、条件付きテストの結果が真になったり、プログラムカウンタが変化したりすると、その命令の実行に 2命令サイクルかかります。この場合、2番目のサイクルはNOPとして実行されます。1命令サイクルは4つのオシレータ周期から成ります。したがって、オシレータ周波数が4MHzの場合、通常の命令実行時間は1μsになります。命令を実行した結果、条件付きテストの結果が真になったり、プログラムカウンタが変化したりすると、命令実行時間は2μsになります。

表7-2に、MPASMアセンブラが認識する命令のリストを示します。

図7-1には、命令が取ることができる3つの一般的なフォーマットを示します。

注意：将来のPIC16CXX製品との上位互換性を維持するために、OPTION命令とTRIS命令を使用しないでください。

本書のすべての例では、次のフォーマットで16進数を表わします。

0xhh

上記フォーマットのhは16進数を表わします。

図7.1　　命令の一般的なフォーマット

バイト対応のファイルレジスタ命令

13　　　　8	7	6　　　　0
OPCODE	d	f(FILE #)

d=0: 宛先レジスタW
d=1: 宛先レジスタf
f: 7ビットのファイルレジスタアドレス

ビット対応のファイルレジスタ命令

13　　　　10	9　　　7	6　　　　0
OPCODE	b(BIT #)	f(FILE #)

b: 3ビットのビットアドレス
f: 7ビットのファイルレジスタアドレス

リテラルおよびコントロール命令

13　　　　8	7　　　　0
OPCODE	k (LITERAL)

k: 8ビットの即値

7.1 命令の説明

ADDLW　　Add Literal to W

Syntax:	[*label*]　ADDLW　k
Operands:	0 ≤ k ≤ 255
Operation:	(W) + k → W
Status Affected:	C, DC, Z
Encoding:	11 | 111x | kkkk | kkkk
Description:	Wレジスタの内容を8ビットのリテラル "k" に加え、この結果をWレジスタに書き込みます。
Words:	1
Cycles:	1
Example:	ADDLW　0x15 命令実行前 W = 0x10 命令実行後 W = 0x25

ANDLW　　AND Literal and W

Syntax:	[*label*]　ANDLW　k
Operands:	0 ≤ k ≤ 255
Operation:	(W) .AND. (k) → W
Status Affected:	Z
Encoding:	11 | 1001 | kkkk | kkkk
Description:	Wレジスタの内容と8ビットのリテラル "k" のANDを取り、結果をWレジスタに書き込みます。
Words:	1
Cycles:	1
Example:	ANDLW　0x5F 命令実行前 W = 0xA3 命令実行後 W = 0x03

ADDWF　　ADD W to f

Syntax:	[*label*]　ADDWF　f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) + (f) → (dest)
Status Affected:	C, DC, Z
Encoding:	00 | 0111 | dfff ffff
Description:	Wレジスタの内容をレジスタ "f" に加え、この結果を、d=0であればWレジスタに、d=1であればレジスタ "f" に書き戻します。
Words:	1
Cycles:	1
Example:	ADDWF　FSR, 0 命令実行前 W = 0x17 FSR = 0xC2 命令実行後 W = 0xD9 FSR = 0xC2

ANDWF　　AND W with f

Syntax:	[*label*]　ANDWF　f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) .AND. (f) → dest
Status Affected:	Z
Encoding:	00 | 0101 | dfff ffff
Description:	Wレジスタとレジスタ "f" のANDを取り、この結果を、d=0であればWレジスタに、d=1であればレジスタ "f" に書き戻します。
Words:	1
Cycles:	1
Example:	ANDWF　FSR, 1 命令実行前 W = 0x17 FSR = 0xC2 命令実行後 W = 0x17 FSR = 0x02

5.2 PORTB

PORTBは８ビット幅の双方向のポート(ファイルレジスタアドレス06h)です。対応するデータ方向レジスタはTRISB(アドレス86h)です。TRISBの１は対応するポートピンを入力としてセットします。PORTBレジスタの呼び出しを行なうとピンのステータスを呼び出しますが、そこに書き込みを行なうとポートラッチに書き込みます。図5.2.1および図5.2.2のPORTBピンのブロック図を参照して下さい。

PORTBピンのそれぞれに内部プルアップ(~100μA平均)があります。ポートピンが出力として構成された場合、そのプルアップは自動的にoffになります。さらに１個のコントロールビットRBPU(OPTION<7>)がすべてのプルアップをoff(RBPUはセット)にすることができます。このプルアップはパワーオンリセットでディスエーブルにされます。

PORTBではそのピン、RB<7:4>において信号変化時の割込みの特長があります。入力として構成されると、これらの入力ピンはサンプリングされ、リードのQ1サイクル上にラッチされます。新しい入力は各命令サイクル毎に前のラッチされた値と比較されます。ピンとラッチ間でミスマッチが起きたときアクティブハイ出力を出します。RB4、RB5、RB6、RB7でのミスマッチ出力は RBIF 割込み(INTCON<0>にラッチ)を発生するためにOR出力されます。出力として構成されたピンは比較からは除かれます。この割込みでチップをSLEEPモードからウエークアップさせることができます。割込みサービスルーチンで、次の2つのうちどれかで割込みをクリアすることができます:

a) RBIE(INTCON<3>)ビットをクリアすることによって割込みをディスエーブルにする。

b) PORTBを読みだす。これによってミスマッチ状態が終わり、RBIFビットをクリアする。

このミスマッチ時の割込みの特長によって、これらの４個のピンのソフトウエアで設定できるプルアップと共にキーパッドとのインターフェースが簡単にでき、キーを押すことによってウエークアップが可能になります。

最後に、ポートピンRB0は外部割込み入力INTと多重化されています。

図5.2.1　　ポートピン RB<7:4>のブロック図

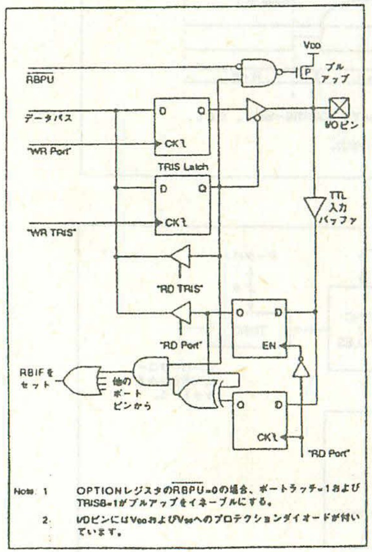
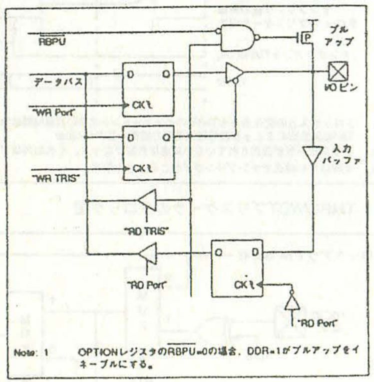


図5.2.2　　ポートピン RB<3:0>のブロック図



Preliminary

表5.2.1　　PORTBの機能

ポートピン	ビット	ピンの機能	別機能
RB0/INT	bit0	入力　出力ポートピン。TTLレベルと内部ソフトウエアプログラマブルなプルアップ	外部割込み入力
RB1	bit1	入力　出力ポートピン。TTLレベルと内部ソフトウエアプログラマブルなプルアップ	
RB2	bit2	入力　出力ポートピン。TTLレベルと内部ソフトウエアプログラマブルなプルアップ	
RB3	bit3	入力　出力ポートピン。TTLレベルと内部ソフトウエアプログラマブルなプルアップ	
RB4	bit4	入力　出力ポートピン。TTLレベルと内部ソフトウエアプログラマブルなプルアップ	ポートでの信号変化時の割込み
RB5	bit5	入力　出力ポートピン。TTLレベルと内部ソフトウエアプログラマブルなプルアップ	ポートでの信号変化時の割込み
RB6	bit6	入力　出力ポートピン。TTLレベルと内部ソフトウエアプログラマブルなプルアップ	ポートでの信号変化時の割込み
RB7	bit7	入力　出力ポートピン。TTLレベルと内部ソフトウエアプログラマブルなプルアップ	ポートでの信号変化時の割込み

表5.2.2　　PORTB レジスタの概要

レジスタの名称	機能	アドレス	パワーオンリセット値
PORTB	リード時PORTBピン ライト時PORTBラッチ	06h	xxxx　xxxx
TRISB	PORTBデータ方向レジスタ	86h	1111　1111
OPTION	弱いプルアップon/offコントロール (RBPUビット)	88h	1111　1111

Preliminary

5.4.1 TIMER0 (TMR0) 割込み

TMR0モジュールタイマ／カウンタがFFhから00hにオーバーフロすると、TMR0割込みが発生します。このオーバーフローが発生すると、TOIFビットがセットされます。この割込みは、TOIEビット（INTCON<5>）をクリアすることでマスクできます。この割込みを再びイネーブルに設定するためには、TMR0モジュールの割込みサービスルーチンを使って、ソフトウェアの中で予めTOIFビット（INTCON<2>）をクリアしておかなければなりません。TMR0タイマはSLEEP中は停止されていますから、TMR0モジュールの割込みが発生してもプロセッサがSLEEP状態からウェークアップすることはありません。図5.4.4にTMR0割込みのタイミングを示します。

5.4.2 外部クロックによるTMR0の使い方

TMR0に外部クロック入力を使ったとき、内部のフェーズクロックと同期を取ることができます。しかし、外部クロック入力は正確に必要な条件を満たさなければなりません。

外部クロックエッジから実際のTMR0のインクリメントまで若干の遅れがあります。図5.4.5を見るとプリスケーラの後に同期が行なわれています。プリスケーラの出力は立ち上がりか立ち下がりエッジを検出するために各命令サイクル毎にサンプリングされています。しかし、tosc＝オシレータタイムの周期とすると、PSOUTには最低2toscの間ハイ、最低2toscの間ローが必要です。

プリスケーラを使用しないときは、PSOUT（プリスケーラ出力、図4.2.1参照）はTMR0クロック入力と同じですが必要なことは：

T_{RTH} ＝ TMR0ハイタイム ≥ 2tosc ＋ ΔT（パラメータ#40参照）

T_{RTL} ＝ TMR0ロータイム ≥ 2tosc ＋ ΔT（パラメータ#41参照）

プリスケーラを使うときは、TMR0モジュール入力は非同期リップルカウンタタイプのプリスケーラで分周されプリスケーラ出力は対称波形となります。

したがって：

T_{RT} ＝ TMR0入力周期

N ＝ プリスケーラ値(2,4,....,256)，とすると

$$P_{SOUT}\text{ハイタイム} = P_{SOUT}\text{ロータイム} = \frac{N \cdot T_{RT}}{2}$$

そこで必要となるのは：

ΔT ＝ 小さなRCディレイ（タイミングの仕様参照）、の時

$$\frac{N \cdot T_{RT}}{2} \geq 2\,t_{osc} + \Delta T \quad \text{または} \quad T_{RT} \geq \frac{4\,t_{osc} + 2\,\Delta T}{N}$$

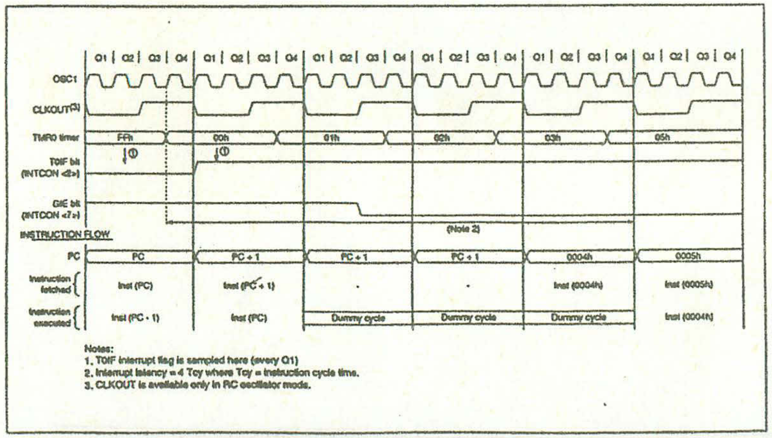
TMR0のハイタイムとロータイムに必要な規格が示されていないことに気がつくと思いますが、TMR0のハイタイムとロータイムは非常に小さいとき、そのパルスを検出できません。最低のハイタイムかロータイムに10ns必要です。TMR0の必要条件をまとめると：

$$T_{RT} = \text{TMR0周期} \geq \frac{4\,t_{osc} + 2\Delta T}{N}$$

T_{RTH} ＝ TMR0ハイタイム ≥ ΔT

T_{RTL} ＝ TMR0ロータイム ≥ ΔT

図5.4.4　TIMER0 (TMR0) 割込みタイミング



Notes:
1. TOIF interrupt flag is sampled here (every Q1)
2. Interrupt latency = 4 Tcy where Tcy = instruction cycle time.
3. CLKOUT is available only in RC oscillator mode.

外部クロックエッジからの遅延：　プリスケーラ出力が内部クロックと同期しているため、外部クロックエッジが起こる時間からTMR0モジュールが実際にインクリメントされる時間までに若干の遅れがあります。図5.4.5を見ると、この遅れ時間が3 toscと7 toscの間にあることがわかります。従って、例えば、2つのエッジの間(周期など)の時間を計ると±4 tosc (±200 ns @ 20 MHz)以内です。

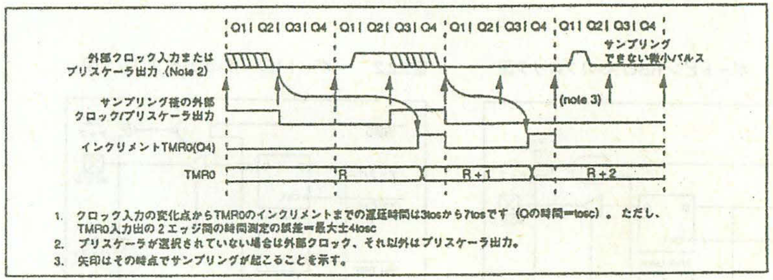
5.4.3 プリスケーラ

1個の8ビットカウンタがTMR0モジュールのプリスケーラとして、または、ウオッチドッグタイマのポストスケーラとしてそれぞれに使用されます（図5.4.6）。このデータシートではわかりやすくするためこのカウンタをプリスケーラとして扱います。TMR0モジュールとウオッチドッグタイマに対し交互に、専用に共有されているプリスケーラが1個しかないことに注意してください。したがって、プリスケーラがTMR0モジューツに設定されている時は、ウオッチドッグタイマ用のプリスケーラがないことになり、反対の場合も同様です。

OPTION<3:0>のPSAとPS2-PS0ビットがプリスケーラの設定とプリスケール比を決めます。

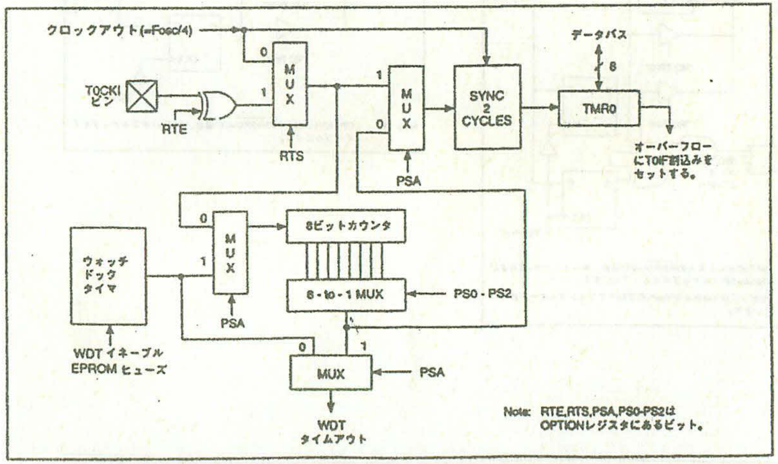
TMR0モジュールに設定されたときは、TMR0モジュールに書き込むすべての命令（CLRF TMR0, MOVWF TMR0, BSF TMR0, x. . . . など）はプリスケーラをクリアします。WDTに設定されたときは、CLRWDT命令がウオッチドッグタイマとともにプリスケーラをクリアします。プリスケーラはリードもライトもできません。

図5.4.5　外部クロックによるTIMER0タイミング



1. クロック入力の変化点からTMR0のインクリメントまでの遅延時間は3toscから7toscです（Qの時間＝tosc）。ただし、TMR0入力出の2エッジ間の時間測定の誤差＝最大±4tosc
2. プリスケーラが選択されていない場合は外部クロック、それ以外はプリスケーラ出力。
3. 矢印はその時点でサンプリングが起こることを示す。

図5.4.6　TMR0/WDTプリスケーラのブロック図



Note:　RTE,RTS,PSA,PS0-PS2はOPTIONレジスタにあるビット。

図4.2.1　パワーアップ時のタイムアウトシーケンス(MCLRとVDD未接続)：ケース1

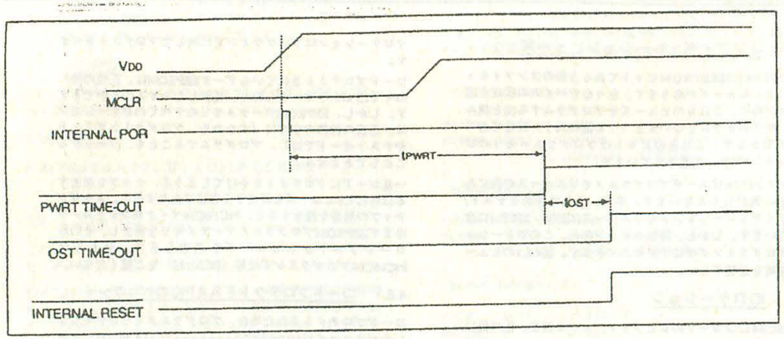


図4.2.2　パワーアップ時のタイムアウトシーケンス(MCLRとVDD未接続)：ケース2

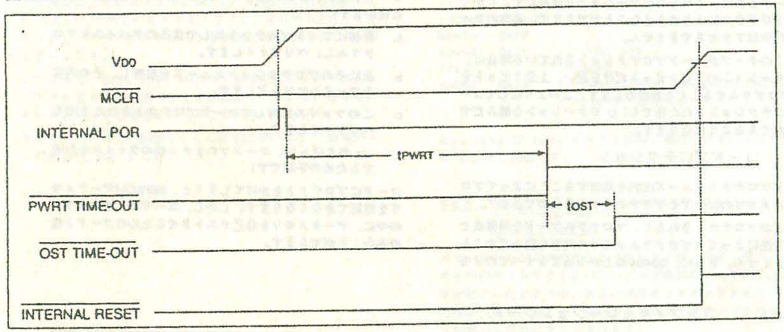


図4.2.3　パワーアップ時のタイムアウトシーケンス(MCLRとVDD接続)：

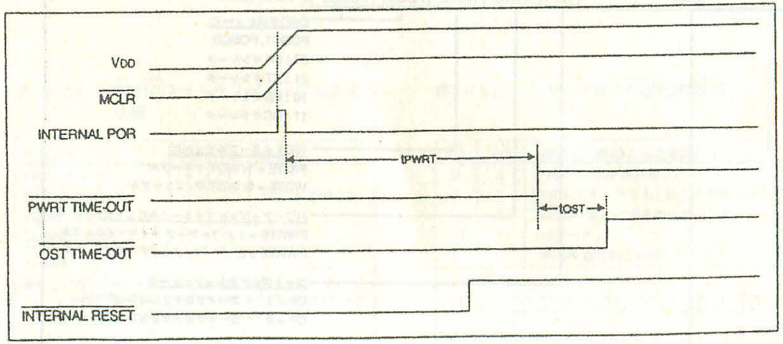
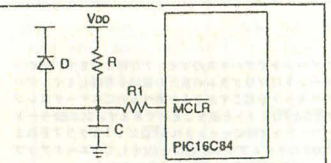
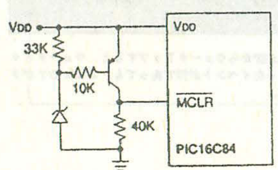


図4.2.4　外部パワーオンリセット回路



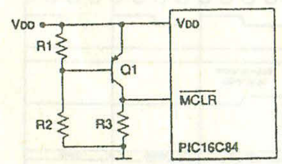
注意：
1. 外部パワーオンリセット回路はVDDのパワーアップ時の立ち上がりがかなり遅い場合のみ必要です。ダイオードDはVDDのパワーダウン時にキャパシタを短時間でディスチャージするのに役立ちます。
2. Rでの電圧降下が0.2V（MCLRピンでの最大漏れ電流の規格が5μA）を確実に超えないようにするためには、R<40K にすることをお薦めします。電圧降下が大きいとMCLRピンのVIHレベルが十分でなくなります。
3. MCLRピンがESDやEOSでブレークダウンした場合、R1(100 −1K)が外部のキャパシタCからMCLRに流れ込む電流を制限します。

図4.2.5　電圧低下保護回路1



注意：
1. VDDが（VZ+0.7V：VZ=ツェナー電圧）以下になったとき、この回路でリセットされる。

図4.2.6　電圧低下保護回路2



注意：
1. この回路は安価ですが正確ではありません。VDDが次の値以下になったときトランジスタQ1がOFFになります：

$$V_{DD} \cdot \frac{R1}{R1 + R2} = 0.7\text{ V.}$$

4.3　ウオッチドッグタイマ(WDT)

外部に部品を必要としない内蔵RCオシレータがフリーランニングしていることによってウオッチドッグタイマが実現されています。つまり、デバイスのOSC1/OSC2ピンにクロックが例えばSLEEP命令によって、停止されても、WDTは動作します。WDTのタイムアウトがデバイスのリセット条件を発生します。コンフィギュレーションヒューズWDTEを0にプログラムすることによって、WDTを永久的にディスエーブルにすることができます(5.6章)。

4.3.1　WDTの周期

WDTの平均タイムアウト周期は18ms（プリスケーラなしで）です。タイムアウト周期は温度、VDD、チップの製造工程によって差があります(DC規格をご覧下さい)。もしさらに長いタイムアウト周期が必要な場合は、OPTIONレジスタに書き込むことによってソフトウエアコントロールにより1:128の分周比のプリスケーラをWDTに設定することができます。このようにして2.5秒までのタイムアウト周期を実現できます。

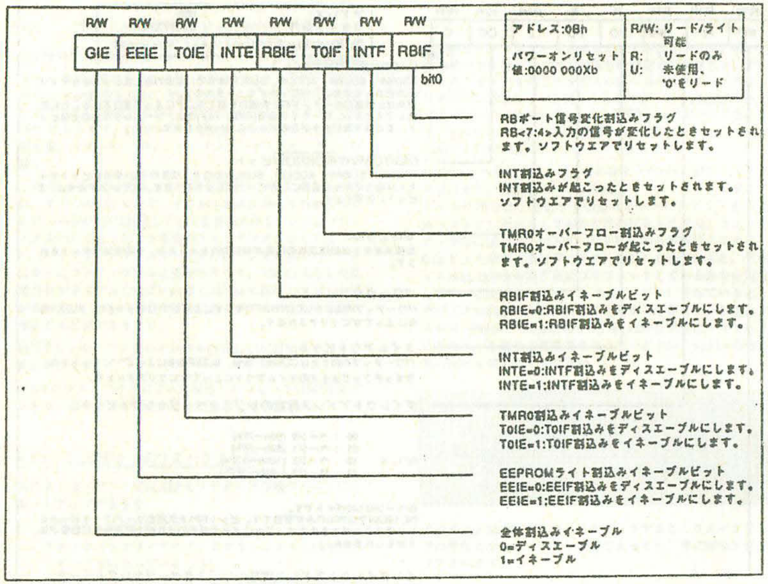
CLRWDTとSLEEP命令が、WDTを指定した場合、WDTとプリスケーラをクリアし、タイムアウトすることとデバイスのリセット条件を発生することを妨げます。

ウオッチドッグタイマがタイムアウトするとSTATUSレジスタのステータスビットTOをクリアします。

4.3.2　WDTプログラミングの注意事項

さらにWDTのタイムアウトになるまでの数秒間に起こる可能性のあるワーストケース(VDD=最小、温度=最高、最大WDTプロスケーラ)を考慮にいれる必要があります。

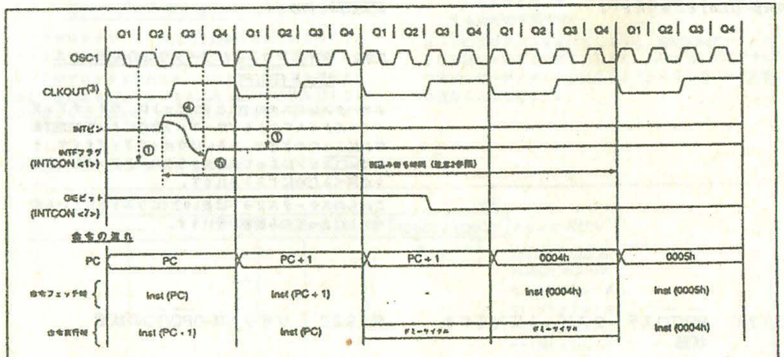
図3.12.3　INTCONレジスタ



3.12.1 INT割込み

RB0/INTピンの外部割込みはエッジトリガです:立ち上がりエッジ（INTEDG=1、OPTION<6>）か、立ち下がりエッジ（INTEDG=0）のどちらか。INTピンに有効なエッジが起こったときは、INTFビットがセットされます（INTCON<1>）。この割込みをINTEコントロールビット（INTCON<4>）でクリアすることによってディスエーブルにできます。この割込みを再びイネーブルにする前に、割込みサービスルーチンのソフトウエアでINTFビット（INTCON<1>）をクリアする必要があります。プロセッサがSLEEPモードになる前にINTEビットがセットしてあれば、INT割込みによってプロセッサをSLEEPモードからウエークアップすることができます。GIEビットのステータスによってプロセッサがウエークアップ後の割込みベクタに分岐するかどうかが決まります。SLEEPについて詳しくは4.5章をご覧ください。SLEEPからウエークアップしてINT割込みに至るタイミングについては図5.5.1を参照してください。

図3.12.1.1　INTピン割込みのタイミング



注意：
1. INTFフラグはここで（Q1ごとに）サンプルされます。
2. 割込み待ち時間=3-4 Tcy（Tcy=割込みサイクル時間）。Inst（PC）がシングルサイクルでも2サイクルでも、待ち時間は同じです。
3. CLKOUTが可能なのはRCオシレータ・モードだけです。
4. INTパルスの最少幅仕様についてはAC仕様を参照してください。
5. INTFはQ4-Q1サイクル中いつでもイネーブルに設定できます。

3.12.2 TMR0割込み

TMR0がオーバーフローすると（FFh → 00h）、T0IF（INTCON<2>）ビットがセットされます。この割込みは、T0IE（INTCON<5>）ビットをセット／クリアすることによってイネーブル／ディスエーブルに設定できます。詳しくは5.4.1章をご覧ください。

3.12.3 PORT RB割込み

PORTB<7:4>の入力が変化すると、RBIF（INTCON<0>）ビットがセットされます。この割込みは、RBIE（INTCON<4>）ビットをセット／クリアすることによってイネーブル／ディスエーブルに設定できます。

3.12.4 EEPROMライト終了時割込み

データEEPROMライトが完了すると、EEPROMライト割込みフラグEEIF（EECON1<4>）がセットされます。この割込みは、EEIEビット（INTCON<6>）をクリアすることによってマスクできます。EEPROMライト割込みについて詳しくは5.6.3章をご覧ください。

3.0 アーキテクチャの概要

PIC16C84 は一般的な RISC タイプマイクロプロセッサの設計思想に見られる多くの特徴を採用しています。まず、PIC16C84 はハーバードアーキテクチャを採用しており、それは、プログラムとデータがそれぞれ別のメモリからアクセスされるようになっています。これによってプログラムとデータが同じメモリからアクセスされるフォンノイマンアーキテクチャよりも実行周期が改善されています。別々のプログラムとデータメモリによってさらに8ビット幅のデータワードと違う幅の命令を持つことができます。PIC16C84 ではオペコードが14ビット幅ですべてシングルワード命令としてもつことができます。14ビット幅のプログラムメモリから14ビット命令をバスにアクセスし、シングルサイクルでそれをフェッチします。2 段のパイプラインによって命令のフェッチと実行を同時に行います。したがって、プログラムの分岐を除いたすべての命令(35個)がシングルサイクルで実行されます。

PIC16C84は内蔵されている1Kx14のプログラムメモリスペースをアドレス指定します。プログラムの実行は内部のみです(マイクロコントローラモード)。

PIC16C84 は48個のレジスタファイルまたはデータメモリを直接、または間接的にアドレス指定できます。プログラムカウンタを含むすべての特殊用途レジスタはデータメモリ内に配置されます。

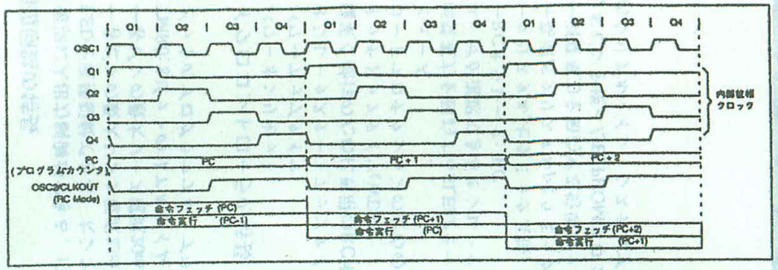
命令セットは非常に直交的（対称的）な構成になっており、この結果、アドレス指定モードやレジスタの種類に係わらず、任意のアドレス指定モードで任意のレジスタから任意の動作を実行することができます。このように命令セットの構成が対称的で、しかも "特定の最適条件" に囚われないため、PIC16C84を使ったプログラミングが簡単にしかも効果的に行え、そのうえ学習効果がさらによくなります。

3.1 PIC16C84 ピン信号の説明

・ピン名称	ピンタイプ	ピン機能：通常動作	ピン機能：シリアルインシステムプログラミング (ISP) モード
VDD	P	電源	電源
Vss	P	グランド	グランド
OSC1/CLKIN	I	クロック入力/オシレータ接続	-
OSC2/CLKOUT	I/O	オシレータ接続/CLKOUT出力 RCオシレータモードではCLKOUT、その他のモードではオシレータ接続	-
$\overline{MCLR}$/VPP	I/P	マスタクリア(外部リセット)入力。アクティブロー	マスタクリア。高電圧 (Vp-p) 印加でプログラミングモードへ。
RA4/T0CKI	I/O	オープンドレーン出力/入力ピン。TMR0タイマ/カウンタへのクロック入力:シュミットトリガ入力バッファ	-
RA0	I/O	双方向I/Oピン。TTL入力レベル	-
RA1	I/O	双方向I/Oピン。TTL入力レベル	-
RA2	I/O	双方向I/Oピン。TTL入力レベル	-
RA3	I/O	双方向I/Oピン。TTL入力レベル	-
RB0/INT	I/O	双方向I/Oピン/外部割込み入力。TTL入力レベル	-
RB1	I/O	双方向I/Oピン。TTL入力レベル	-
RB2	I/O	双方向I/Oピン。TTL入力レベル	-
RB3	I/O	双方向I/Oピン。TTL入力レベル	-
RB4	I/O	双方向I/Oピン。TTL入力レベル	-
RB5	I/O	双方向I/Oピン。TTL入力レベル	-
RB6	I/O	双方向I/Oピン。TTL入力レベル	クロック入力
RB7	I/O	双方向I/Oピン。TTL入力レベル	クロック入力／出力

説明: I＝入力、O＝出力、I/O＝入力／出力、P＝電源、-＝未使用.

図 3.2.1 クロック/命令サイクル



3.2 クロック波形/命令サイクル

クロック入力(OSC1ピンから)は内部で4分周され、Q1、Q2、Q3、Q4と呼ばれる4個のオーバラップしない矩形波クロックを出力します。内部で、PCがQ1毎にインクリメントされ、命令がプログラムメモリからフェッチされQ4で命令レジスタにラッチされます。クロックと命令の実行フローを図3.2.1に示します。

3.3 命令のフロー/パイプライン動作

命令サイクルはQ1、Q2、Q3、Q4でできています。命令のフェッチと実行サイクルはフェッチに1命令サイクル、デコードと実行にほかの命令サイクルがかかるようにパイプライン方式で行われます。しかしパイプライン方式によって、それぞれの命令は効率よく1サイクルで実行します。もし命令によってプログラムカウンタが変化した場合(GOTOなど)は、その命令を完全に実行するためには2サイクルが必要です。

フェッチサイクルはQ1でプログラムカウンタ(PC)のインクリメントと同時に始まります。

フェッチされた命令は命令レジスタ(IR)にラッチされて、Q2、Q3、Q4の間にデコードされ実行されます。データメモリに対してはQ2の間リードが行われ(オペランドリード)、Q4の間ライトが行われます(ディスティネーションライト)。

3.4 プログラムメモリ構成

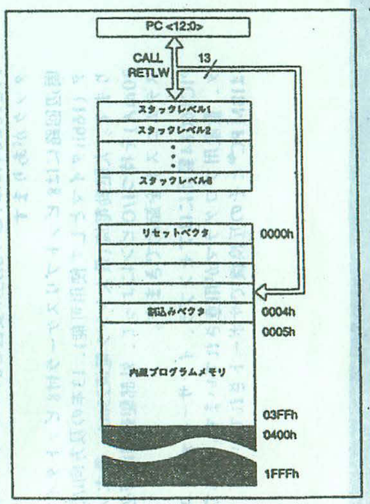
PIC16C84には8K×14のプログラムスペースをアドレス指定できる13ビット幅のプログラムカウンタがあります。(図3.4.1) 最初の幅の1K×14(0000h-03FFh)のみ物理的に設定されています。3FFh以上のロケーションをアクセスするとその最初の1K×14のスペース内で循環することになります。リセットベクタは0000hに、割り込みベクタは0004hにあります。

PIC16C84のEEPROMプログラムメモリは、イレース／ライトサイクルが可能な設定になっています。プログラムメモリをプログラムしたいときには、まずMCLRピンをハイに上げて特殊モードをオンにしてください（プログラミングの仕様については6.0章をご覧ください）。また、プログラミング中は、VDDを4.5Vから5.5Vに保つ必要があります。

PIC16C84は、ユーザのアプリケーションによってプログラムメモリが頻繁に更新されるような用途には適しません。

プログラムメモリは2つのデータ／クロックピンを使って逐次的にプログラムすることができ（6.0節参照）、これを行うとインシステムプログラミング（ISP）が可能になります。インシステムプログラミングを利用すると、最終試験中にシステムをカスタマイズしたり、現場でシステムをアップグレードしたりすることができます。

図 3.4.1 プログラムメモリマップとスタック



MICROCHIP

PIC16C84

8-Bit CMOS EEPROMマイクロコントローラ

特長

RISC-likeな高性能CPU
- 覚える必要があるのは35個のシングルワード命令のみ
- 2サイクルのプログラム分岐を除いて、全てシングルサイクル(400ns)
- 動作スピード: DC-10MHz クロック入力
  DC-400ns 命令サイクル
- 14ビット幅の命令
- 8ビット幅のデータバス
- 1024×14ビットの内蔵EEPROMプログラムメモリ
- 36×8ビットの汎用レジスタ(SRAM)
- 15個の特殊用途ハードウエアレジスタ
- 64×8ビットEEPROM データメモリ
- 8レベルのハードウエアスタック
- ダイレクト(直接)、インダイレクト(間接)、リラティブ(相対)の各アドレスモード
- 4個の割り込み要因:
  —外部 INT ピン
  —TMR0タイマオーバーフロー時の割込み
  — PORTB<7:4> 信号変化時の割込み
  —データEEPROMライト終了時の割込み
- 1,000,000回のERASE/WRITE サイクル（標準）
- データ保持期間40年以上

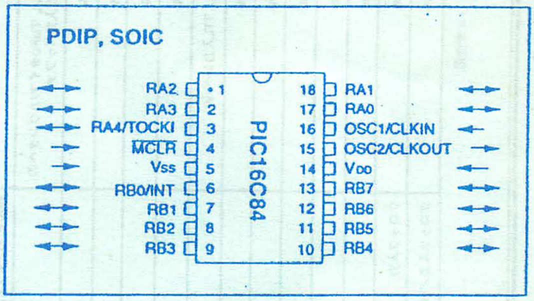
周辺回路の特長

- 個別に入出力制御ができる、13本のI/Oピン
- LEDを直接駆動できる、大シンク/ソース電流
  —各ピンの最大シンク電流25mA
  —各ピンの最大ソース電流20mA
- TMR0: 8ビットのリアルタイム・クロック カウンタ（8ビットのプログラマブル・プリスケーラ付き）

マイクロコントローラの特長

- パワーオンリセット
- パワーアップタイマ
- オシレータスタートアップタイマ
- 確実な動作のために専用のRCオシレータを内蔵した、ウォッチドッグタイマ(WDT)
- コードプロテクションのための、セキュリティEEPROMヒューズ
- 消費電力を節約するSLEEPモード
- ユーザが選択できるオシレータオプション:
  —RCオシレータ: RC
  —クリスタル/セラミック共振: XT
  —高周波クリスタル/セラミック共振: HS
  —消費電力を節約する低周波クリスタル: LP
- 2本ピンを使ったEEPROMプログラムおよびデータメモリのシリアル・イン・システムプログラミング(ISP)

図A　　ピン配置



CMOSテクノロジー

- 高速、低消費電力CMOS EEPROMテクノロジー
- 完全スタティック設計
- 余裕の動作電圧範囲
  —商用: 2.0V - 6.0V
  —工業用: 2.0V - 6.0V
- 低消費電力
  —<2mA @ 5V、4MHz
  —60µA平均 @ 2V、32KHz
  —25µA平均スタンバイ電流 @ 2V

概要

PIC16C84は高性能、低価格、CMOS、完全スタティック、8ビットセキュリティのマイクロコントローラで、1Kx14のEEPROMプログラムメモリと64バイトのEEPROMデータメモリを内蔵しています。このマイクロコントローラは、高性能を誇るPIC16CXXファミリの2番目の製品です（PIC16C5X製品をご使用中のユーザは付録Aのリストをご覧ください）。

新しいPIC16C84は、プログラム分岐以外のすべての命令をシングルワード（14ビット幅）とし、各命令をシングルサイクル（10MHzで400ns）で実行することによって高性能を実現しています。プログラム分岐には2サイクル（800ns）が必要です。

PIC16C84には4個の割込み要因と8レベルのハードウエアスタックがあります。

周辺回路には8ビットプリスケーラ付8ビットタイマ/カウンタ（16bitタイマとして使用可能）、13本の双方向I/Oピンがあります。大駆動電流（最大シンク電流25mA、最大ソース電流20mA）を持つI/Oピンによって、外部駆動回路が必要なくシステムコストを節約できます。

PIC16C84製品にはアセンブラ、インサーキット・エミュレータ、量産用プログラマが用意されています。すべてのツールはIBM PC®とその互換機でサポートされています。

Ｃ５コンデンサは極性がありますので基板シルク印刷の＋側にあわせます。Ｃ６コンデンサは極性はありません。

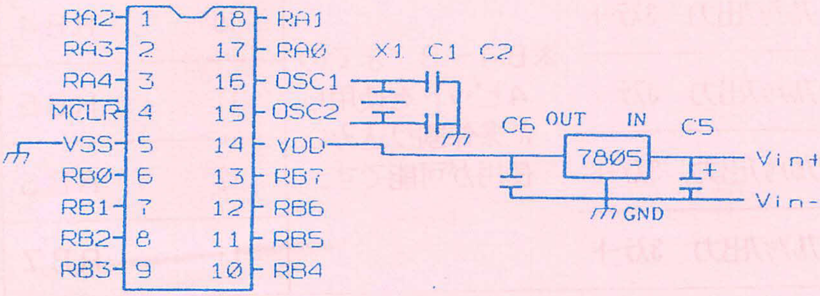
③発振用クリスタルＸ１、コンデンサＣ１，Ｃ２を取り付けます。クリスタルは10MHzが付属していますが、システムにあわせて４MHz～10MHzのクリスタルが使用可能です。

※クリスタルはサイズの都合上、他の部品にぶつかる恐れがありますので、必ず最後に取り付けるようにして下さい。

※基板上にＵ２，Ｘ２，Ｃ３，Ｃ４がありますがここには何も取り付けません。

《回路図》

Ｕ１　ＰＩＣ１６Ｃ８４





《Ｉ／Ｏコネクタピン（Ｊ１）の説明》　～その１～

IOピン	I/O	機　能	ICピンNo.	ICピン名称
ＭＣＬ	Ｉ	マスタークリア，外部リセット，アクティブＬｏｗ	４	ＭＣＬＲ
ＶＤＤ	Ｐ	電源５Ｖ	14	ＶＤＤ
ＧＮＤ	Ｐ	グランド	５	ＶＳＳ
ＡＯ	I/O	ＴＴＬレベル入力　Ｃ－ＭＯＳレベル出力	17	ＲＡＯ
Ａ１	I/O	ＴＴＬレベル入力　Ｃ－ＭＯＳレベル出力	18	ＲＡ１
Ａ２	I/O	ＴＴＬレベル入力　Ｃ－ＭＯＳレベル出力	１	ＲＡ２
Ａ３	I/O	ＴＴＬレベル入力　Ｃ－ＭＯＳレベル出力	２	ＲＡ３
ＣＣ	I/O	シュミットトリガ入力　オープンドレイン出力（カウンタタイマ用入力機能ピン）	３	RA4/TOCKI

I=Input　O=Output　P=Power（電源，ＧＮＤ）

《I/Oコネクタピン（J1）の説明》 ～その2～

IOピン	I/O	機能	ICピンNo.	ICピン名称
B0	I/O	TTLレベル入力　プルアップ出力　3ステート（外部割り込み入力）	6	RB0/INT
B1	I/O	TTLレベル入力　プルアップ出力　3ステート	7	RB1
B2	I/O	TTLレベル入力　プルアップ出力　3ステート	8	RB2
B3	I/O	TTLレベル入力　プルアップ出力　3ステート	9	RB3
B4	I/O	TTLレベル入力　プルアップ出力　3ステート	10	RB4
B5	I/O	TTLレベル入力　プルアップ出力　3ステート	11	RB5
B6	I/O	TTLレベル入力　プルアップ出力　3ステート	12	RB6
B7	I/O	TTLレベル入力　プルアップ出力　3ステート	13	RB7
---	I	水晶発振端子／水晶発振入力／クロック入力	16	OSC1/CLKIN
---	O	水晶発振端子／水晶発振出力／クロック出力	15	OSC2/CKOUT

※B4～B7までの4ビットを使用した条件割り込みに使用が可能です。

I=Input　O=Output　P=Power（電源，GND）

※IOピンとは基板上のシルク印刷の事を指します。

※シルク印刷上ではIOピンが2箇所存在しますが、'AKIPICプラチナキット'ではシルク印刷側からみて右側部分（J1）を使用します。左側部分（J2）は使用しません。

当キットのハードウエアに関する御質問は封書か往復葉書でお願い致します。

AKIPICプラチナキットマニュアル 10MHz版　　第1版　H07.03.05
秋月電子通商　〒158　東京都世田谷区瀬田5-36-6

by KAKE & Jふえ Special thanks for 岸ちゃん

PIC16C84のプログラム書込に関する参考資料

※PIC16C84のプログラム書込に関する技術資料及び書込回路例，プログラム例を添付致しますので、今後のプログラムにお役立てください。

《PIC16C84ライタの特徴》

☆パソコンパラレルポート（セントロニクスプリンタポート）を使用するため、簡単なソフト、ハードになります。

☆クロック、データ（送受各１），５Ｖ制御Ｖｐｐ制御の５種類の信号で出来ます。（シリアルモード時）

☆EEPROMのプログラムエリア部、及びデータエリア部にリード、ライト、イレースが可能です。

☆電源電圧　13.5V

MICROCHIP

PIC16C84

PIC16C84 EEPROM Memory Programming Specification

PROGRAMMING THE PIC16C84

The PIC16C84 is programmed using one of two methods, serial or parallel. The serial mode will allow the PIC16C84 to be programmed while in the users system. This allows for increased design flexability. The parallel mode will provide faster programming as the data is loaded into the PIC16C84 with a greater throughput. Either mode may be selected at the start of the programming process.

Hardware Requirements

The PIC16C84 requires two programmable power supplies, one for VDD (4.5V to 5.5V) and one for VPP (12V to 14V). Both supplies should have a minimum resolution of 0.25V.

Programming Mode

The programming mode for the PIC16C84 allows programming of user program memory, data memory, special locations used for ID, and the configuration fuses for the PIC16C84.

1.0 PROGRAM MODE ENTRY

TABLE 1.0 - TEST MODE SELECTION

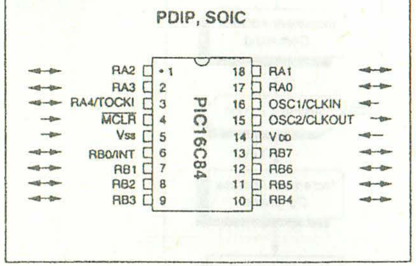
RB7 RB6 RB5 RB4	Mode	Function
0 0 X X	TMOD1	Program/verify

1.1 User Program Memory Map

The user memory space extends from 0000h to 1FFFh (8K), of which 1K (0000h - 03FFh) is physically implemented. In actual implementation the on-chip user program memory is accessed by the lower 10-bits of the PC, with the upper 3-bits of the PC ignored. Therefore if the PC is greater than 3FFh, it will wrap around and address a location within the physically implemented memory. (See Figure 1.1.1).

In programming mode the program memory space extends from 0000h to 3FFFh, with the first half (0000h-1FFFh) being user program memory and the second half (2000h-3FFFh) being configuration memory. The PC will increment from 0000h to 1FFFh to 2000h to 3FFFh and wrap around to 2000h (not to 0000h). Once in configuration memory, the highest bit of the PC stays a '1', thus always pointing to the configuration memory. The only way to point to user program memory is to reset the part and reenter program/verify mode (TMOD1) as described in Section 1.2.

FIGURE A - PIN CONFIGURATION



In the configuration memory space, 2000h-200Fh are physically implemented. Locations beyond 200Fh will physically access user memory. (See figure 1.1.1).

1.2 TMOD1: Program/Verify Mode

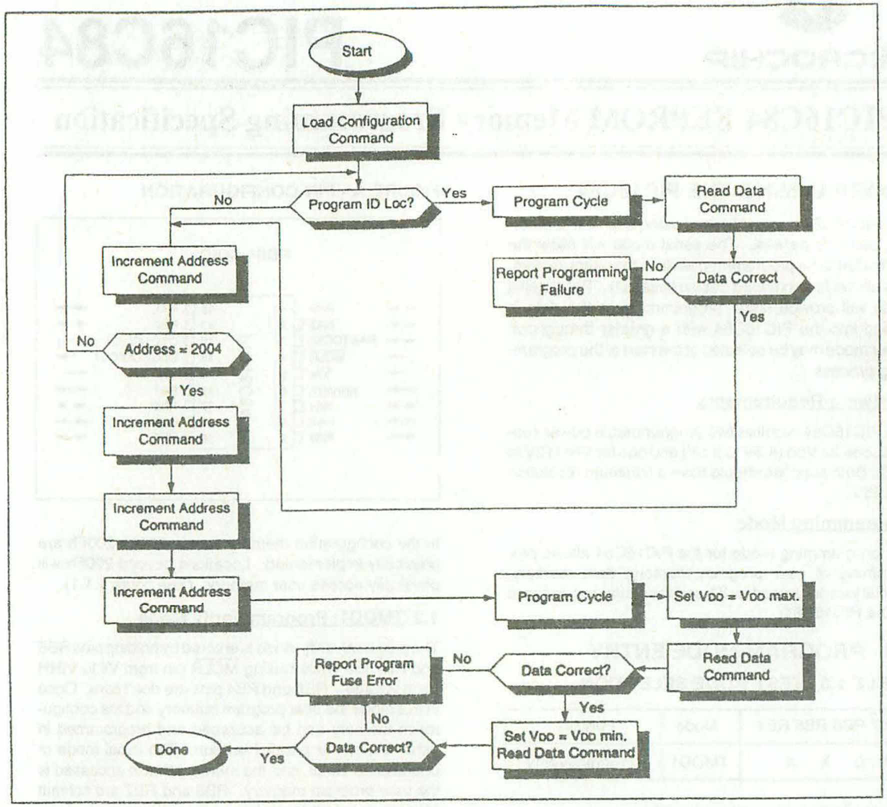
The program/verify mode is entered by holding pins RB6 and RB7 low while raising MCLR pin from Vil to ViHH (high voltage). RB5 and RB4 pins are don't care. Once in this mode the user program memory and the configuration memory can be accessed and programmed in either a serial or parallel fashion. The initial mode of operation is serial, and the memory that is accessed is the user program memory. RB6 and RB7 are schmitt trigger inputs in this mode.

The sequence that enters the device into the program-ming/verify mode places all other logic into the reset state (the MCLR pin was initially at VIL). This means that all I/O are in the reset state (High impedance inputs).

1.2.1 SERIAL PROGRAM/VERIFY OPERATION

The RB6 pin is used as a clock input pin, and the RB7 pin is used for entering command bits and data input/output during serial operation. To input a command, the clock pin (RB6) is cycled six times. Each command bit is latched on the falling edge of the clock with the least significant bit (lsb) of the command being input first. The data on pin RB7 is required to have a minimum setup and hold time (see AC/DC specifications) with respect to the falling edge of the clock. Commands that have data associated with them (read and load) are specified to have a minimum delay of us between the command and the data. After this delay, the clock pin is cycled 16 times with the first cycle being a start bit and the last cycle being a stop bit. Data is also input and output lsb first.

FIGURE 1.2.1.2 - PROGRAM FLOW CHART - PIC16C84 CONFIGURATION MEMORY



1.2.1.3 Load Data for Data Memory

After receiving this command, the chip will load in 14-bits as a "data word" when 16 cycles are applied. However, the data memory is only 8-bits wide, and thus only the first 8-bits of data after the start bit will be programmed into the data memory. It is still necessary to cycle the clock the full 16 cycles in order to allow the internal circuitry to reset properly. The data memory contains 64 words. Only the lower 8-bits of the PC are decoded by the data memory, and therefore if the PC is greater than 3Fh, it will wrap around and address a location within the physically implemented memory.

1.2.1.4 Read Data from Program Memory

After receiving this command, the chip will transmit data bits out of the program memory (user or configuration) currently accessed starting with the second rising edge of the clock input. The RB7 pin will go into output mode on the second rising clock edge, and it will revert back to input mode (hi-impedence) after the 16th rising edge. A timing diagram of this command is shown in Figure 1.2.1.4.

1.2.1.5 Read Data from Data Memory

After receiving this command, the chip will transmit data bits out of the data memory starting with the second rising edge of the clock input. The RB7 pin will go into output mode on the second rising edge, and it will revert back to input mode (hi-impedence) after the 16th rising edge. As previously stated, the data memory is 8-bits wide, and therefore, only the first 8-bits that are output are actual data.

1.2.1.6 Increment Address

The PC is incremented when this command is received. A timing diagram of this command is shown in Figure 1.2.1.5.

FIGURE 1.2.1.3 - LOAD DATA FOR PROGRAM MEMORY COMMAND (SERIAL PROGRAM/VERIFY)

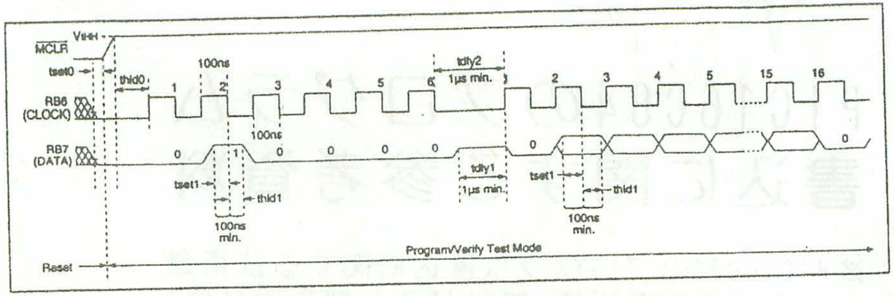


FIGURE 1.2.1.4 - READ DATA FROM PROGRAM MEMORY COMMAND (SERIAL PROGRAM/VERIFY)

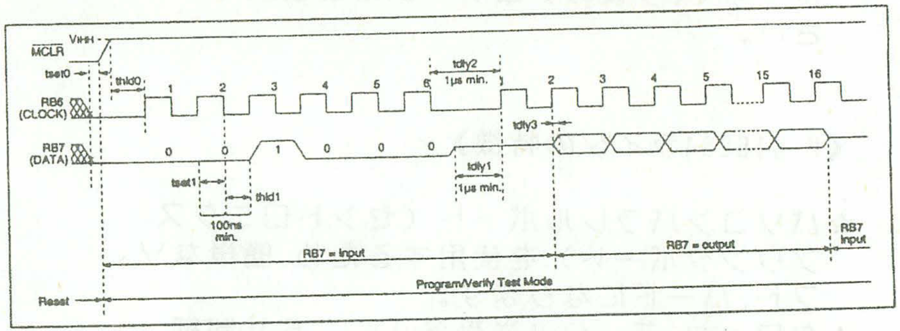
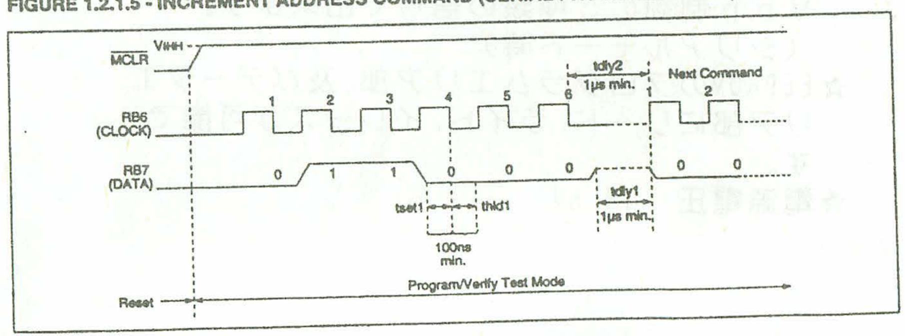


FIGURE 1.2.1.5 - INCREMENT ADDRESS COMMAND (SERIAL PROGRAM/VERIFY)



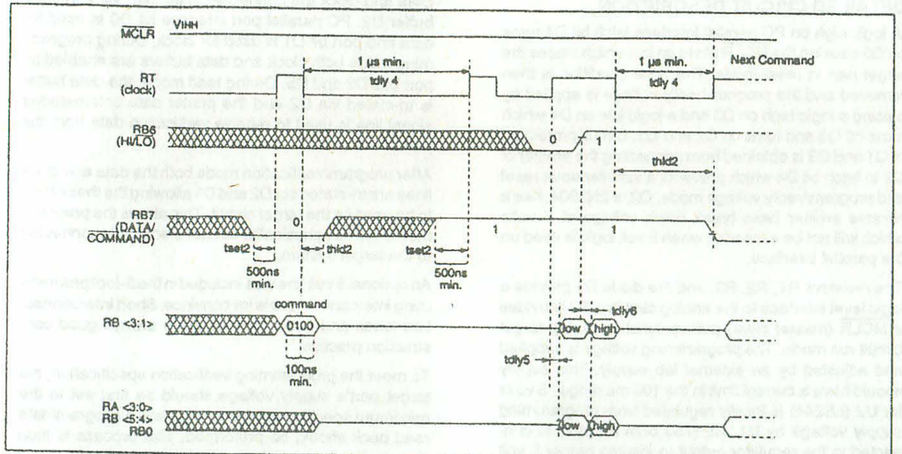
1.2.1.7 Begin Programming

A load command must be given before every begin programming command. Programming of the appropriate memory (test program memory, user program memory or data memory) will begin after this command is received and decoded. An internal timing mechanism executes an erase before write. The user must allow 10ms for programming to complete. No "end programming" command is required.

1.2.1.8 Enter Parallel Mode

After receiving this command, the chip will accept commands and supply data in a parallel fashion. After parallel mode has been entered, serial operation can only be achieved by fully exiting the program/verify mode and re-entering. See Section 1.2.2 for details.

FIGURE 1.2.2.3 - READ DATA FROM PROGRAM MEMORY COMMAND (PARALLEL PROGRAM/VERIFY)



1.2.2.7 Begin Programming

A load command must be given before every begin programming command. Programming of the selected memory (configuration memory, user program memory or data memory) will begin after this command is received and decoded. The programming time is specified to be 10ms. The OSC2 pin will go high when the programming is completed (only in parallel mode).

1.2.2.8 Bulk Erase Program Memory

After this command is performed, the next program command will erase the entire program memory. The erase time is specified to be 10ms.

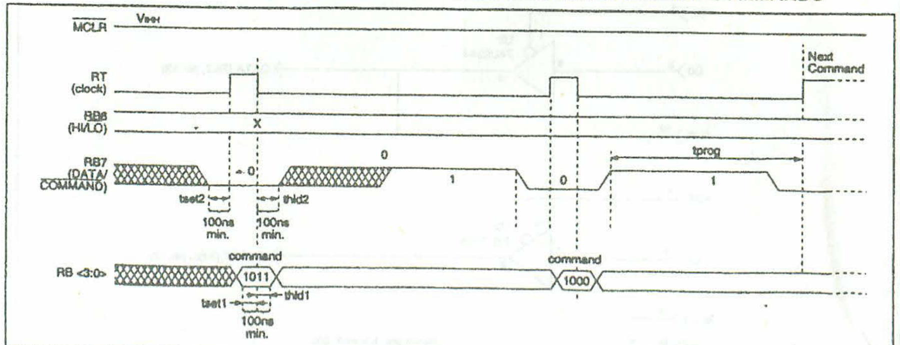
If the address is pointing to user memory, only the user memory will be erased.

If the address is pointing to the test program memory (2000h - 200Fh), then both the user memory and the test memory will be erased. The configuration word will not be erased, even if the address is pointing to location 2007h.

1.2.2.9 Bulk Erase Data Memory

After this command is performed, the next program command will erase the entire data memory. The erase time is specified to be 10ms. A description of the sequence of these two commands is shown in Figure 1.2.2.4.

FIGURE 1.2.2.4 - BULK ERASE DATA MEMORY AND BEGIN PROGRAMMING COMMANDS



1.3.1 AC/DC TIMING REQUIREMENTS FOR PROGRAM/VERIFY TEST MODE

AC/DC CHARACTERISTICS, POWER SUPPLY PINS	Standard Operating Conditions Operating temperature 0 ≤ TA ≤ +70°C, unless otherwise stated Operating voltage 4.5V ≤ VDD ≤ 5.5V, unless otherwise stated					
Characteristic	**Sym**	**Min**	**Typ**	**Max**	**Units**	**Conditions/Comments**
General						
Supply voltage during programming	VDDP	4.5	5.0	5.5	V	
Supply voltage during verify	VDDV	VDD min.		VDD max.	V	Note 1
High voltage on MCLR and RT for test-mode entry	VIHH	12		14.0	V	Note 2
Supply current (from VDD) during program/verify	IDDP			50	mA	
Supply current from VIHH (on MCLR)	IHH			1	mA	
MCLR rise time (VSS to VHH) for test mode entry	tVHHR			1.0	µs	
(RB6, RB7) input high level	VIH1	0.8 VDD			V	Schmitt trigger input
(RB6, RB7) input low level	VIL1	0.2 VDD			V	Schmitt trigger input
RB<7:4> setup time before MCLR↑ (test mode selection pattern setup time)	tset0	100			ns	
RB<7:4> hold time after MCLR↑ (test mode selection pattern hold time)	thld0	100			ns	
Serial Program/Verify						
Data in setup time before clock↓	tset1	100			ns	
Data in hold time after clock↓	thld1	100			ns	
Data input not driven to next clock input (delay required between command/data or command/command)	tdly1	1.0			µs	
Delay between clock↓ to clock↑ of next command or data	tdly2	1.0			µs	
Clock↑ to data out valid (during read data	tdly3	80			ns	
Parallel Program/Verify						
Data in setup time before clock↓	tset3	1			µs	
Data in hold time after clock↓	thld3	1			µs	
RB6 and RB7 setup time before clock↓	tset1	1			µs	
RB6 and RB7 hold time after clock↓	thld1	1			µs	
RT (clock)↓ to RT (clock)↑	tdly4	2			µs	
RB7 (data/command select input) setup before RT (clock)↑	tset2	1			µs	
RB7 (data/command select input) hold time after RT (clock)↓	thld2	1			µs	
RT (clock)↑ to data out valid	tdly5	1			µs	

(cont.)

MICROCHIP

AN589

A PC-Based Development Programmer for the PIC16C84

Author: Robert Spur - Analog Design Specialist, Inc.

PROGRAMMING THE PIC16C84 MICROCONTROLLER

This application note describes the construction of a low cost serial programmer for the PIC16C84 microcontroller which is controlled using a PC with a parallel (Centronix printer) port. This programmer has the capability of programming the PIC16C84 microcontroller, and reading back internal data without removing the device form the target circuit.

This feature is very useful in applications where changes in program code or constants are necessary to compensate for other system features. For example, an embedded control system may have to compensate for variances in a mechanical actuator performance or loading. The basic program can be programmed and tested in design. The final program and control constants can be easily added later in the production phase without removing the microcontroller from the circuit.

Automatic software and performance upgrades can also be implemented with an in-system. Upon receiving new system software via disk or modem, a control processor with the included programming code could perform an in circuit reprogramming of other microcontrollers in the system.

This programmer can load program code, part configuration, and EEPROM data into the PIC16C84 part. In read back mode, it can verify all verify all data entries.

FUNCTION DESCRIPTION

The PIC16C84 microcontroller is put into programming mode by forcing a low logic level on the RB7 (pin 13) and RB6 (pin 12) while the MCLR (pin 4) is first brought low to reset the part, and then brought to the program/verify voltage of 12 to 14 volts. The MCLR pin remains at the program/verify voltage for the remainder of the programming or verification time.

After entering programming mode, RB7 is used to serially enter programming modes and data into the part. A high to low transition on RB6, the clock input, qualifies each bit of the data applied on RB7. The serial command-data format is specified in Figure 1.2.1.3 of the Microchip PIC16C84 Programming Specification (DS30189D). The first 6 bits form the command field, and the last 16 bits form the data field. Notice that the data field is composed of one zero starting bit, 14 actual data bits, and one zero stop bit. The increment address command, shown in Figure 1.2.1.5 (see PIC16C84 Data Sheet, DS30189D), is comprised of only the command field. Table 1.2.1.1 (see DS30189D) summarizes the available commands and command codes for serial programming mode.

The read mode is similar to programming mode with the exception that the data direction of RB7 is reversed after the 6-bit command to allow the requested data to be returned to the programmer. Figure 1.2.1.4 (see DS30189D) shows this sequence which starts by shifting the 6-bit command into the part. After the read command is issued, the programmer tri-states its buffer to allow the part to serially shift its internal data back to the programmer. The rising edge of RB6, the clock input, controls the data flow by sequentially shifting previously programmed or data bits from the part. The programmer qualifies this data on the falling edge of RB6. Notice that 16 clock cycles are necessary to shift out 14 data bits.

Accidental in circuit reprogramming is prevented during normal operation by the MCLR voltage which should never exceed the maximum circuit supply voltage of 6V DC and the logic levels of port bits RB7 and RB8.

After program/verification the MCLR pin is brought low to reset the target microcontroller and electrically released. The target circuit is then free to activate the MCLR signal. In the event MCLR is not forced by the target circuit, R4 (a 2K Ohm pull up resistor in the programmer) provides a high logic level on the target microcontroller which enables execution of its program independent of the programmer connection. Provisions should be made to prevent the target circuit from resetting the target microcontroller with MCLR or effecting the RB6 and RB6 during the programming process. In most cases this can be done without jumpers.

DETAILED CIRCUIT DESCRIPTION

A logic high on PC parallel interface latch bit D4 turns on Q3 causing the MCLR pin to go low which places the target part in reset mode. The reset condition is then removed and the program/verify voltage is applied by placing a logic high on D3 and a logic low on D4 which turns off Q3 and turns on Q2 and Q1. Circuit protection of Q1 and Q3 is obtained from connecting the emitter of Q2 to latch bit D4 which prevents a simultaneous reset and program/verify voltage mode. Q2, a 2N3904, has a reverse emitter base break down voltage of 6 volts which will not be exceeded when 5 volt logic is used on the parallel interface.

The resistors R1, R2, R3, and the diode D1 provide a logic level interface to the analog circuitry. R4 provides a MCLR (master clear) pull up function during target circuit run mode. The programming voltage is supplied and adjusted by an external lab supply. This supply should have a current limit in the 100 ma range. 5 volts for U2 (LS244) is locally regulated from programming supply voltage by U1. R5 (750 ohm resistor) is connected to the regulator output to insures proper 5 volt regulation when the 13.5 volt programming voltage is applied through the pull up resistor R4.

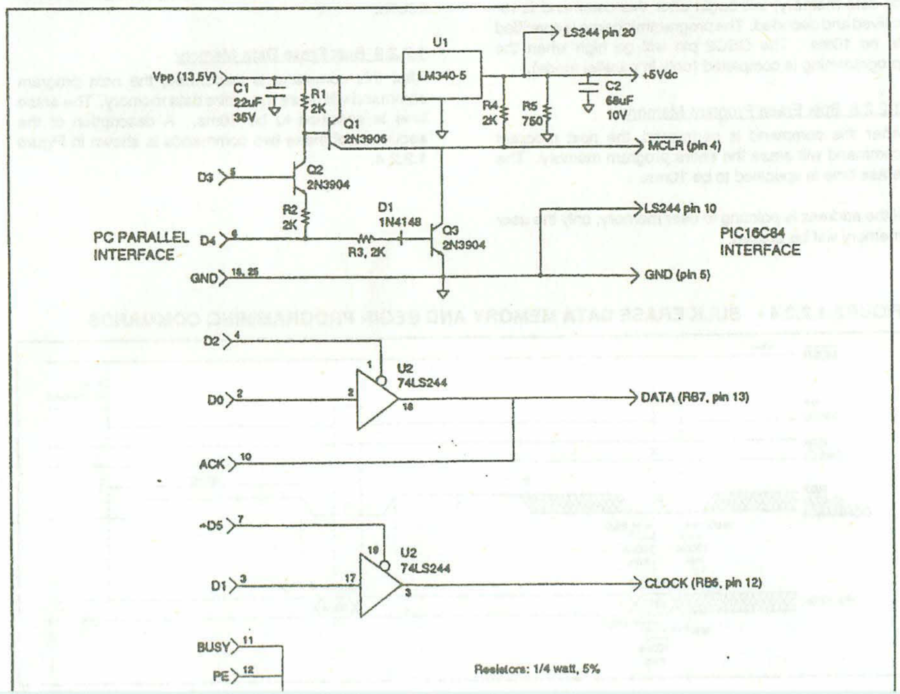
Data and clock are connected to the part via a tri-state buffer U2. PC parallel port interface bit D0 is used for data and port bit D1 is used for clock. During programming mode both clock and data buffers are enabled by port bits D2 and D5. During read mode, the data buffer is tri-stated via D2 and the printer data acknowledge signal line is used to receive verification data from the part.

After program/verification mode both the data and clock lines are tri-stated via D2 and D5 allowing the these lines to be used by the target circuit. This allows the programmer to remain physically, but not electrically connected to the target system.

An optional 5 volt line was included in the 3-foot programming interconnect cable for convince. Short interconnection leads and good grounding are always good construction practice.

To meet the programming verification specification, the target part's supply voltage should be first set to the maximum specified supply voltage and a program/data read back should be preformed. This process is then repeated at the lowest specified supply voltage.

FIGURE 1: PROGRAMMER SCHEMATIC



```
s_cmd = cmd;                                   // retain command "cmd"
for (i=0;i<6;i++)                              // output 6 bits of command
  {
  biosprint(0,(s_cmd&0x1) +2+8,PTR);           // set bits 001010, clock hi
  biosprint(0,(s_cmd&0x1)   +8,PTR);           // set bits 001000, clock low
  s_cmd >>=1;
  }

if((cmd ==INC_ADDR)||(cmd ==PARALLEL_MODE)     // command only, no data cycle
  return 0;

else if(cmd ==BEGIN_PROG)                      // program command only, no data cycle
  {
  delay(10);                                   // 10 ms PIC programming time
  return 0;
  }

else if((cmd ==LOAD_DATA)||(cmd ==LOAD_DATA_DM)||(cmd ==LOAD_CONFIG)) // output 14 bits of
data
  {
  for (i=200;i;i--) ;                          // delay between command & data
  biosprint(0,2+8,PTR);                        // set bits 001010, clock hi; leading bit
  biosprint(0,  8,PTR);                        // set bits 001000, clock low

  for (i=0;i<14;i++)                           // 14 data bits, lsb first
   {
   biosprint(0,(data&0x1) +2+8,PTR);           // set bits 001010,  clock hi
   biosprint(0,(data&0x1)   +8,PTR);           // set bits 001000,  clock low
   data >>=1;
   }
  biosprint(0,2+8,PTR);                        // set bits 001010, clock hi; trailing bit

// ***************** Analog Design Specialists *****************

  biosprint(0,  8,PTR);                        // set bits 001000, clock low

  return 0;
  }

 else if((cmd ==READ_DATA)||(cmd ==READ_DATA_DM)) //read 14 bits from part, lsb first
  {
  biosprint(0,  4+8,PTR);                      // set bits 001100, clock low, tri state data
                                                          buffer
  for (i=200;i;i--) ;                          // delay between command & data
  biosprint(0,2+4+8,PTR);                      // set bits 001110, clock hi, leading bit
  biosprint(0,  4+8,PTR);                      // set bits 001100, clock low

  data =0;
  for (i=0;i<14;i++)                           // input 14 bits of data, lsb first
   {
   data >>=1;                                  // shift data for next input bit
   biosprint(0,2+4+8,PTR);                     // set bits 001110,  clock hi
   biosprint(0,  4+8,PTR);                     // set bits 001100,  clock low
   if(!(biosprint(2,0,0)&0x40)) data += 0x2000; //use printer acknowledge line for input,
                                                          data lsb first
   }
  biosprint(0,2+4+8,PTR);                      // set bits 001110, clock hi, trailing bit
  biosprint(0,  4+8,PTR);                      // set bits 001100, clock low
  return data;
  }

 else return PIC_PROG_EROR;                    // programmer error

 }
else if(cmd == RESET)                          // reset device
 {
 biosprint(0,32+16+4,PTR);                     // set bits 110100, MCLR = low (reset
 delay(1);                                     // 1ms delay
 biosprint(0,32   +4,PTR);                     // set bits 100100, MCLR = high
 return 0;
 }

else if(cmd ==PROGRAM_MODE)                    // enter program mode
 {
 biosprint(0,32+16+4,PTR);                     // set bits 110100, Vpp off, MCLR = low
                                                    (reset PIC16C84)
 delay(10);                                    // 10 ms, allow programming voltage to
                                                    stabelize

 biosprint(0,8,PTR);                           // set bits 001000, Vpp on , MCLR = 13.5
                                                    volts, clock & data connected
 delay(10);                                    // 10 ms, allow programming voltage to
                                                    stablize

 return 0;
 }

else if(cmd ==RUN)                             // disconects programmer from device
 {
 biosprint(0,32+4,PTR);                        // set bits 100100
 return 0;
 }
else return PROGMR_ERROR;                      // command error
}
```

A PC-Based Development Programmer for the PIC16C84

登録商標

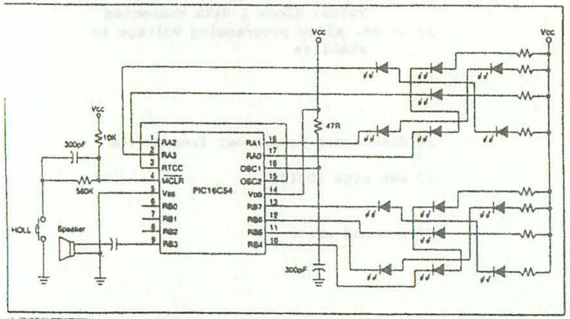
PICはMicrochip Technology Incorporated in the U.S.A.の登録商標です。

Microchipのロゴおよび名称はMicrochip Technology Incorporatedの登録商標です。

Dice Game

AUTHOR: David Beecher
Kirkland, WA

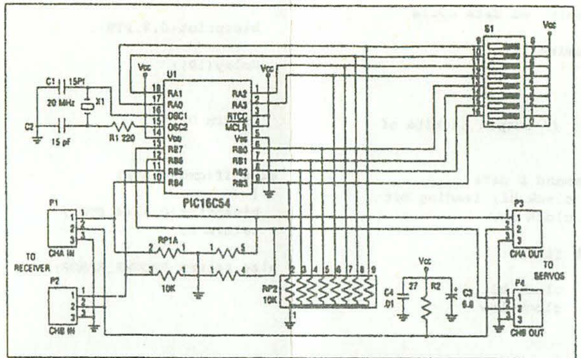
This design rolls two digital dice as part of a game. The reset is used to wake up the PIC16C54 microcontroller to begin the dice roll. The speaker makes some sound effects as the dice is rolling. After the dice stop rolling, the result is displayed for a while, the LEDs are turned off and the processor goes back to sleep. Because of the PIC16C54's low current during sleep, no power switch is required to turn it off. If designed using TTL, eight chips would be required just to run the die (no sound). The design will fit in one PLD but a clock would still be required, no sound, and no sleep mode. If battery powered, it would have to be turned off when not in use. The design features a unique reset circuit that will send out one short reset pulse no matter how long reset is pressed.



Radio Control Aircraft

AUTHOR: Edward F. Castro
Acworth, GA

This design implements a low cost, versatile airborne control mixer for radio control model aircraft. Four modes are provided including one and two way mixing, rate selection, and exponential control. In the past, these advanced features were available only by investing in a computer radio costing several hundreds of dollars. By running the PIC16C54 at 20 MHz, a resolution of +/- 100 parts can be realized which translates to less than 1/2 degree at the servo output arm. Target price for this unit is $39.95 list. Street price could be as low as $30.00 or about the price of a standard servo. At these prices, the modeler can afford to have a unit installed in each of several aircraft.

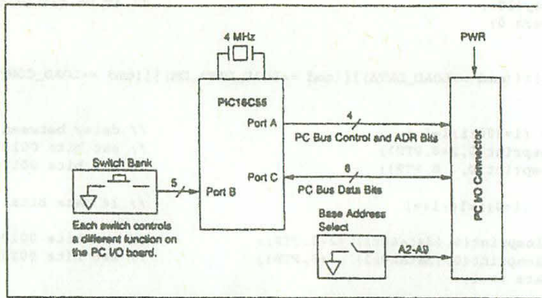


Emulating a Personal Computer (PC) with PIC16C55

AUTHOR: Gary C. Sutcliffe
Slinger, WI

I produce and sell an IBM® personal computer (PC) interface card. To demo the unit, I needed to drag along a complete PC, something that is inconvenient. Now I use a PIC16C55 microcontroller to emulate a PC. When the PIC16C55 detects a switch closed, it drives the PC bus signals as if a PC was controlling the interface. This PC simulator is great for debugging units on the bench, too. You avoid the wear and tear on the PC from powering it up and down, as well as saving time waiting for it to boot. Essentially, this design replaces a $1500 PC with a $5 PIC16C55 and comes in a more portable package! The submitted program is of course specific to my application, but could be modified to control other PC I/F cards. This could be used in stand-alone systems with low cost PC I/F cards where a PC host wouldn't be cost effective.

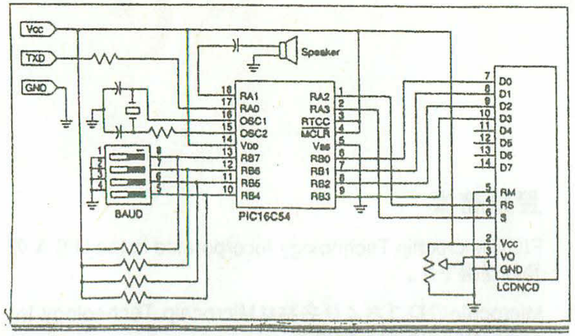
USING A PIC16C55 TO CONTROL A PC INTERFACE CARD



Serial Controlled LCD Module

AUTHOR: David Beecher
Kirkland, WA

Most standard LCD modules have a slow parallel interface. In order to connect to the computers data bus, wait states need to be generated to support the display. This makes the computer slow and requires additional hardware. Since most computer systems have serial interfaces, a serial-to-parallel interface could be used to connect the display. This design uses a PIC16C54 microcontroller to convert the parallel display to an RS232 serial interface. Serial data enters directly into the PIC16C54 microcontroller. The internal clamp diodes of the PIC16C54 and the external resistor allow RS232 levels to be used with no problem. Most displays use very little power. With the PIC16C54's low power consumption, this circuit could steal power directly from the serial port to operate. The dip switches control baud rate. If baud is not a concern, the pins could be hooked to the display module to run it in 8-bit mode rather than 4-bit as shown.

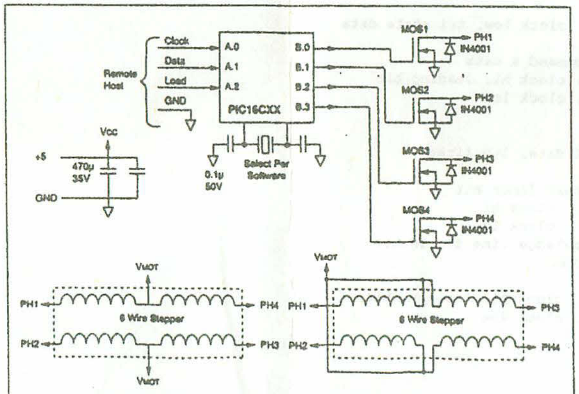


Stepper Motor Controller

AUTHOR: Christopher L. Eddy
Irwin, PA

A PIC16CXX-based stepper motor controller accepts a serial position command. Designed to interface with 6-wire and 8-wire motors. In half and full step driving modes. Simple MOSFET driver stage drives motor directly from DC supply, and 5V only operation is possible. Fifteen parts are required, including motor.

STEPPER MOTOR DRIVER



Size VMOT to correspond to stepper max. winding voltage.
IF VMOT = Vcc, decouple generously.
Size MOS1 - 4 for 2x winding current, typical choice is IRF540.

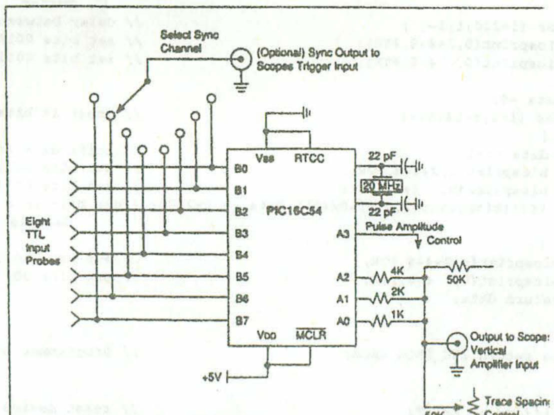
Eight Channel Logic Analyzer

AUTHOR: Jerry Farmer
Myriad Development Co.
Lakewood, CO

Real time display of eight TTL lines on any standard scope - an Eight Channel Logic Analyzer. Design converts digital inputs to time domain using a PIC16/17 microcontroller, a simple 1-2-4 resistor ladder D/A converter and a scopes sweep.

Use .5 volts/division on the scopes vertical sensitivity with DC coupling. Use trace spacing control to separate the eight channels on the display. Adjust the Pulse Amplitude control to effect amplitude of all signals.

EIGHT CHANNEL LOGIC ANALYZER



* Copy circuit for sixteen TTL Logic Inputs on Dual Scopes.

Personal Computer (PC) Link

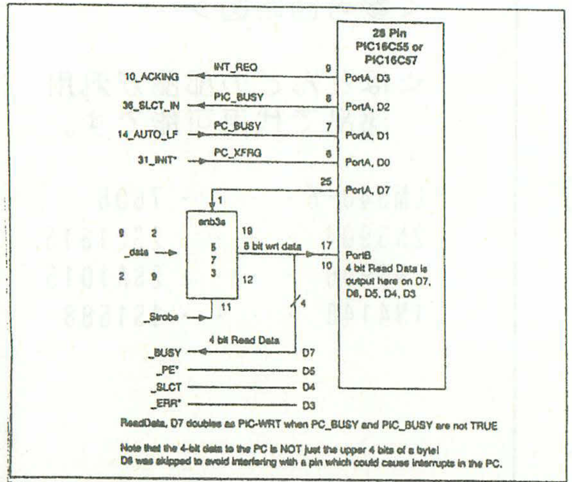
Digital Oscilloscope

"LPT2PIC". This is a link from a personal computer (PC) to the serial world of PIC16CXX. It uses the PC's printer port since, in our present days of network printer sharing, these ports are often times unused. The PC usually refers to its printer connection as LPT, hence the name. The difficulty in this connection has to do with the fact that the PC's printer port is not a complete parallel port. It transfers 8 bits in one direction and only a few status bits in the other.

The second 90 percent of this effort was figuring out what the PIC16CXX microcontroller side of the link should look like. I was disappointed when I realized that Microchip's Application Note AN515 was only good for point-to-point IIC. To really make PIC16CXX talk to an IIC "BUS", an extra open-collector chip and two additional PIC16CXX pins are required. Wanting to conserve precious pins, I designed a two pin, 3-wire protocol (two signals plus ground) which better suits the natural abilities of PIC16CXX.

AUTHOR: Frank Harvey
Davis, CA

"LPT2PIC" HARDWARE SCHEMATIC



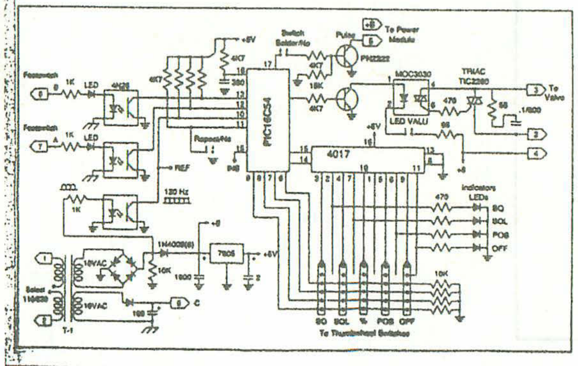
ReadData, D7 doubles as PIC-WRT when PC_BUSY and PIC_BUSY are not TRUE

Note that the 4-bit data to the PC is NOT just the upper 4 bits of a byte! D6 was skipped to avoid interfering with a pin which could cause interrupts in the PC.

Speed Control for Welding Transformer

Controller for a Spot-Welding Transformer. Reads times set in thumbnail switches and send the proper power by phase control.

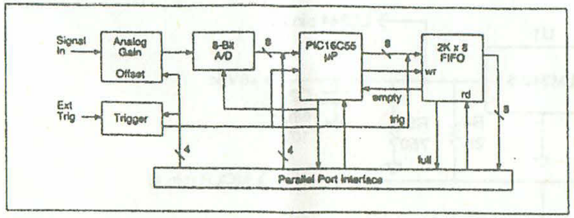
AUTHOR: Ruben De La Pena
Monterrey, N.L. Mexico



This application is a digital oscilloscope that connects to the parallel port of a personal computer (PC). This samples at 200 KHz and provides 3db response at 50 KHz. When triggered, the PIC16C55 reads the A/D converter every 5 ms and accumulates the minimum and maximum values in each sample period. These are then written to 2K x 9 FIFO which is read by the PC. The PC then takes each minimum/maximum pair and draws a vertical line between those values at each horizontal pixel position. In other words, a 10 ms sweep that displays on one-half of a VGA screen would require 320 minimum/maximum pairs per .01 second (31 µs per pair). The display then shows the information from 5 µs sampling as well as it can be displayed at 31 µs per horizontal pixel. This provides maximum resolution in a simple oscilloscope display. The circuit has two triggers: one internal or external, one A/C or level. It also has adjustable gain and offset to handle 0.1 volts to 25 volts full scale.

AUTHOR: Ben Allgor
Paw Paw, MI

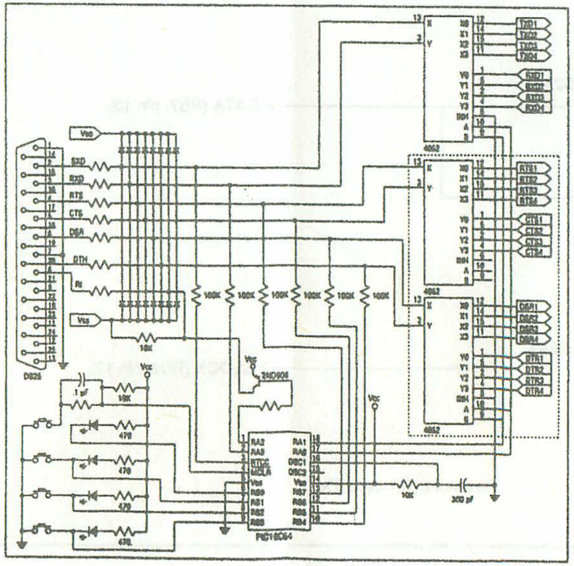
PARALLEL PORT OSCILLOSCOPE



Serial Communication Switch

This design enables a serial port to be connected to four serial devices with the ability to be electronically switched. The four buttons select which port to use. Button 1 also functions as a reset in case of lockup. Because of the PIC16C54 low current, power to run this circuit can be derived from the serial lines. Analog switches are used to switch the lines - they are low cost and use very little power. Some of the incoming signals can be input into the PIC16C54 for checking port activity. These inputs can signal the user of data by blinking the selected ports LED. The reset circuit is unique as it only supplies a short reset pulse no matter how long the reset button is pressed.

AUTHOR: David Beecher
Kirkland, WA

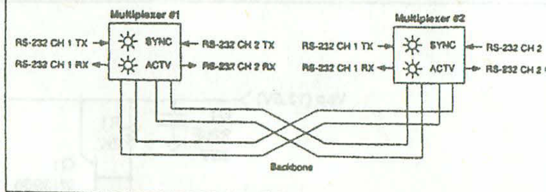


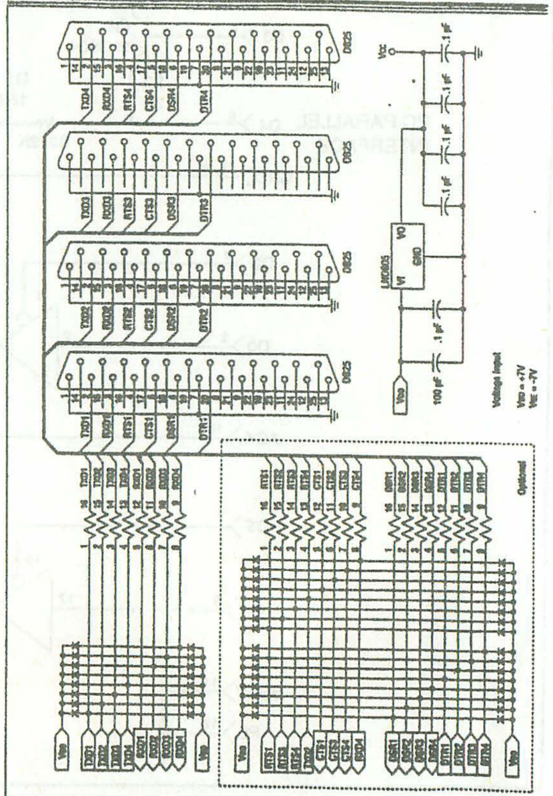
Dual RS-232 Port Multiplexer

This entry is a design (hardware and firmware) for a dual RS-232 port multiplexer. Each multiplexer contains two 2-wire asynchronous communication channels. The channels can be run at a rate of up to 9600 BPS using a 19.6608 MHz processor. Both channels can run full duplex and at independent rates. Two multiplexers connect together using a 4-wire RS-232 interface and normally provide communications between channel 1 on each unit and channel 2 on each unit. The destination of data received by the multiplexer on each RS-232 channel can be set to any of the other three ports in system via a keystroke sequence. There are two LE provided which indicate RS-232 activity and that the t multiplexers are communicating.

AUTHOR: Bill Parton
Cocoa, FL

DUAL PORT MULTIPLEXER SYSTEM BLOCK DIAGRAM

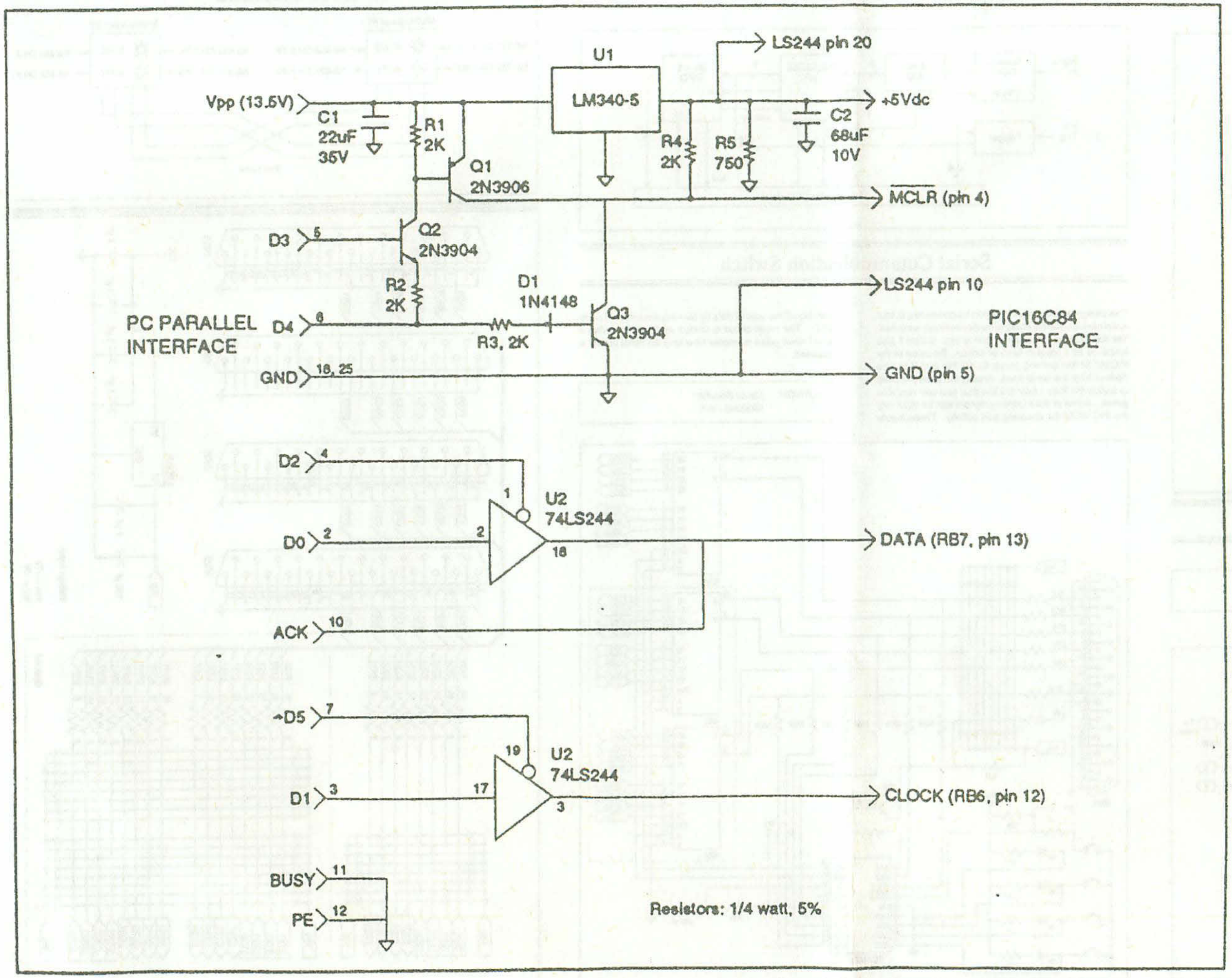




ＰＩＣシリーズ応用回路例です。１６Ｃ８４でも使用可能なものもあります。

PIC16C84), programmer not conected

FIGURE 1: PROGRAMMER SCHEMATIC



＜参考回路図＞

☆ほとんどの部品が汎用
部品で代用可能です。

LM340-5 ・・・・ 7805
2N3904 ・・・・ 2SC1815
2N3906 ・・・・ 2SA1015
1N4148 ・・・・ 1S1588

A PC-Based Development Programmer for the PIC16C84

SOFTWARE DESCRIPTION

The listed code provides a hardware-software interface to a standard PC parallel (Centronix) interface port. The code can be adapted to a microprocessor parallel interface port by substituting an output command for the "biosprint" command.

Control software can transfer the PIC16C84 program, configuration bits, and EEPROM data from a standard PROM interface file into the target system by reading the file and calling the function in Figure 2 using the appropriate command name in the definition table, and the data to be programmed. The command names are repeated here for reference.

LOAD_CONFIG	Sets PIC16C84 data pointer to configuration.
LOAD_DATA	Loads, but does not program, data.
READ_DATA	Reads data at current pointer location.
INC_ADDR	Increments PIC16C84 data pointer.
BEGIN_PROG	Programs data at current data pointer location.
PARALLEL_MODE	Puts PIC16C84 into parallel mode. (not used)
LOAD_DATA_DM	Loads EEPROM data.
READ_DATA_DM	Reads EEPROM data.

Function "int ser_pic16c84(<command>,<data [or 0]>) is called to preform command. Function returns internal data after read commands.

Do not forget to initiate the programming mode before programming, increment the addresses after each byte is programmed, and put the programmer in run mode after programming.

Designed by: Analog Design Specialist, Inc.
P.O. Box 26-0846
Littleton, CO 80126

EXAMPLE 1: PUT TARGET SYSTEM INTO PROGRAM MODE.

```
.. program code..
ser_pic16c84(PROGRAM_MODE,0);
.. program code..
```

EXAMPLE 2: READ DATA FROM THE TARGET SYSTEM

```
.. program code..
data = ser_pic16c84(READ_DATA,0); // read data
// transfers data from target part to variable "data".
.. more program code ..
```

EXAMPLE 3: PROGRAM DATA INTO THE TARGET SYSTEM

```
.. program code..
ser_pic16c84(LOAD_DATA,data);     // load data into target
ser_pic16c84(BEGIN_PROG,0);       // program loaded data
ser_pic16c84(INC_ADDR,0);         // increment to next address
 // transfers data from program variable "data" to target
    part.
.. more program code ..
```

EXAMPLE 4: PUT TARGET SYSTEM INTO RUN MODE

```
.. program code..
ser_pic16c84(RUN,0);
.. program code..
```

```
//************************** FIGURE #2 *********************************
//**                                                                  **
//**    SERIAL PROGRAMMING ROUTINE FOR THE PIC16C84 MICROCONTROLLER   **
//**                                                                  **
//**                    Analog Design Specialists                     **
//**                                                                  **
//**********************************************************************

//FUNCTION PROTOTYPE: int ser_pic16c84(int cmd, int data)

// cmd: LOAD_CONFIG    -> part configuration bits
//      LOAD_DATA      -> program data, write
//      READ_DATA      -> program data, read
//      INC_ADDR       -> increment to the next address (routine does not auto increment)
//      BEGIN_PROG     -> program a previously loaded program code or data
//      LOAD_DATA_DM   -> load EEPROM data regesters (BEGIN_PROG must folow)
//      READ_DATA_DM   -> read EEPROM data
//
// data: 1) 14 bits of program data or
//       2)  8 bits of EEPROM data (least significant 8 bits of int)

// Additional programmer commands (not part of PIC16C84 programming codes)
//
// cmd: RESET          -> provides 1 ms reset pulse to target system
//      PROGRAM_MODE   -> initializes PIC16C84 for programming
//      RUN            -> disconnects programmer from target system
//
// function returns:1) 14 or 8 bits read back data for read commands
//                  2) zero                       for write data commands
//                  3) PIC_PROG_EROR = -1 for programming function errors
//                  4) PROGMR_ERROR  = -2 for programmer function errors


#include <bios.h>

#define LOAD_CONFIG    0
#define LOAD_DATA      2
#define READ_DATA      4
#define INC_ADDR       6
#define BEGIN_PROG     8
#define PARALLEL_MODE 10  // not used
#define LOAD_DATA_DM   3
#define READ_DATA_DM   5
#define MAX_PIC_CMD   63  // division between pic and programmer commands

#define RESET         64  // external reset command, not neded for programming
#define PROGRAM_MODE  65  // initialize program mode
#define RUN           66  // electrically disconnect programmer

#define PIC_PROG_EROR -1
#define PROGMR_ERROR  -2

#define PTR            0  // use device #0

// parallel port bits
//      d0: data output to part to be programmed
//      d1: programming clock
//      d2: data direction, 0= enabel tri state buf -> send data to part
//      d3: Vpp control 1= turn on Vpp
//      d4: ~MCLR =0, 1 = reset device with MCLR line
//      d5: clock line tri state control, 0 = enable clock line

int ser_pic16c84(int cmd, int data)        // custom interface for pic 16c84
  {
  int i, s_cmd;

  if(cmd <=MAX_PIC_CMD)                    // all programming modes
    {
    biosprint(0,8,PTR);                    // set bits 001000, output mode, clock & data low
```

1.3.1 AC/DC TIMING REQUIREMENTS FOR PROGRAM/VERIFY TEST MODE (Cont.)

AC/DC CHARACTERISTICS, POWER SUPPLY PINS	Standard Operating Conditions Operating temperature $0 \le T_A \le +70°C$, unless otherwise stated Operating voltage $4.5V \le V_{DD} \le 5.5V$, unless otherwise stated					
Characteristic	**Sym**	**Min**	**Typ**	**Max**	**Units**	**Conditions/Comments**
Parallel Program/Verify (cont.)						
RB6 (hi/lo select) valid to data out valid	tdly6	100			ns	
Programming pulse width	tprog	10			ms	
Time delay from program to compare (HV discharge time)	tdis	0.5			µs	

Note 1: Program must be verified at the minimum and maximum VDD limits for the part.
Note 2: VIHH must be higher than VDD + 4.5V to stay in programming/verify mode.

2.0 CONFIGURATION FUSES

The PIC16C84 has five configuration fuses, which are EEPROM bits. These fuses can be programmed (reads '0') or left unprogrammed (reads '1') to select various device configurations.

FIGURE 2.0.1 - CONFIGURATION WORD

12 … 5	CP	PWRTE	WDTE	FOSC1	FOSC0 (bit0)
	CP	PWRTE	WDTE	FOSC1	FOSC0

Addr: 2007h

OSC selection fuses:
FOSC1, FOSC0:
00 : LP oscillator
01 : XT oscillator
10 : HS oscillator
11 : RC oscillator

WDT enable fuses:
WDTE = 1: WDT enabled
WDTE = 0: WDT disabled

Power-up timer enable fuse:
PWRTE = 1 power-up time enabled
PWRTE = 0 power-up timer disabled

Code protection fuse:
CP = 1 code protection off
CP = 0 code protection on

Unimplemented. Read as '1's.

2.1 Embedding Configuration Fuse and ID Information In the Hex File

2.2 Code Protection

The code in the program memory and data in the data memory can be protected by programming the code protect fuse (CP).

When code protected, the contents of the program memory cannot be read out in a way that the program code can be reconstructed. It is also not possible to read out the contents of the data memory. In addition, it is not possible to program any memory locations (data or program) while code protection is on.

2.2.1 VERIFYING A CODE-PROTECTED PIC16C84

When code protected verifying any program memory location will read a scrambled output which looks like "0000000xxxxxxx" (binary) where X is 1 or 0. To verify a device after code protection, follow this procedure:

a. First, program and verify a good device without code protecting it.
b. Next, blow its code protection fuse and then load its contents in a file.
c. Verify any code-protected PIC16C84 against this file.

Date memory can not be verified after code protection.

2.3 ID Locations

Four memory locations (2000h - 2003h) are designated as ID locations where the user can store checksum or other code - identification numbers. These locations are not accessible during normal execution but are readable and writable during program/verify.

It is recommended that the user use only the lower 7-bits of the ID locations and always program the upper 7-bits as '1's.

2.2.2 DISABLING CODE-PROTECTION

It is recommended that the following procedure be performed before any other programming is attempted. It is also possible to turn code protection off (code protect fuse = 1) using this procedure; however, all data within the program memory and the data memory will be erased when this procedure is executed, and thus, the security of the data or code is not compromised.

Procedure to disable code protect:

a. Execute load configuration (with a '1' in bit 4, code protect).
b. Increment to fuse location (2007h)
c. Execute command (0001 - RB<3-0>)
d. Execute command (0111 - RB<3-0>)
e. Execute 'Begin Programming'
f. Wait 10ms
g. Execute command (0001 - RB<3-0>)
h. Execute command (0111 - RB<3-0>)

3.0 EMBEDDING DATA EEPROM CONTENTS IN HEX FILE

The programmer should be able to read data EEPROM information from a hex file and conversely (as an option) write data EEPROM contents to a hex file along with program memory information and fuse information.

The 64 data memory locations are logically mapped starting at address 2100h.

4.0 PROGRAMMING ALGORITHM REQUIRES VARIABLE VDD

The PIC16C84 uses an intelligent algorithm. The algorithm calls for program verification at VDD (min.) as well as VDD (max.). Verification at VDD (min.) guarantees good "erase margin". Verification at VDD (max) guarantees good "program margin".

The actual programming must be done with VDD in the VDDP range (4.5 - 5.5V).

VDDP = Vcc range required during programming.
VDD min. = minimum operating VDD spec for the part.
VDD max. = maximum operating VDD spec for the part.

Programmers must verify the PIC16C84 at its specified VDD max. and VDD min. levels. Since Microchip may introduce future versions of the PIC16C84 with a broader VDD range, it is best that these levels are user selectable (defaults are ok). **Any programmer not meeting these requirements may only be classified as "prototype" or "development" programmer but not a "production" quality programmer.**

1.2.2 PARALLEL OPERATION

After receiving the command 'Enter Parallel Mode', the circuitry will function in a parallel fashion. The function of pins RB6 and RB7 change during parallel operation, and it is also necessary to control several more pins to achieve the desired results. Since the program memory in this device is 14-bits wide and there are only 12 I/O pins, it is necessary to transfer data into and out of the chip in two data words. These high and low data words are selected based upon the value of pin RB6 (RB6 = '1': high data word). In parallel mode, both the high and low segments of data input/output are 10-bits wide, making it possible to handle up to 20-bit data words. The need for 20-bit capability is because future versions of PIC16C84 may need to read/program data words larger than 14-bit.

For each 10-bit data-segment (high or low), RB0 is the lsb and RA3 is the msb. In PIC16C84, the lower 14 bits of the total 20-bit word is actually used. The upper-six bits are don't care (see Figure 1.2.2.1). The pin RB7 also has a new function in parallel operation and that is to indicate whether a command is being input or data is being transferred (RB7 = '1' - data transfer). Commands are input on pins RB<3:0> with the lsb applied to pin RB0. The clock pin during parallel operation is the RT pin. Data is again transmitted on the rising edge and latched on the falling edge of the clock. A minimum setup and hold time of 100ns with respect to the falling edge of the clock is again required. When entering a command RB7 must be held low for a minimum of 100ns after the falling edge of the clock. The commands which are available during parallel operation are shown in Table 1.2.2.1.

Note: [illegible] DATA/COMMAND pin (RB7). [illegible] command the RB7 [illegible] back to back commands are being executed without any data word between them.

TABLE 1.2.2.1 - COMMAND MAPPING (PARALLEL OPERATION)

Command	Mapping (RB<3:0>)	Data
Load Configuration	0 0 0 0	Yes
Load Data for Program Memory	0 0 1 0	Yes
Read Data from Program Memory	0 1 0 0	Yes
Increment Address	0 1 1 0	No
Begin Programming	1 0 0 0	No
Read Data from Data Memory	0 1 0 1	Yes
Load Data from Data Memory	0 0 1 1	Yes
Bulk Erase Program Memory	1 0 0 1	No
Bulk Erase Data Memory	1 0 1 1	No

1.2.2.1 Load Configuration

After receiving this command, the program counter (PC) will be set to 2000h pointing to the first location in the configuration memory. By then setting pin RB7 high and pin RB6 high, the upper 4-bits of the complete data word will be latched in on the falling edge of the RT pin. Setting pin RB6 low allows the lower 10-bits to be latched in, and then the entire data word can be programmed into the configuration memory.

Note: The data must be loaded high byte first and then low byte.

A description of the memory mapping schemes for normal operation and programming mode operation is shown in Figure 1.0. After the configuration memory is entered, the only way to get back to the user program memory is by exiting the program/verify mode.

1.2.2.2 Load Data for Program Memory

After receiving this command, the RB7 pin is set high, and the data can then be latched into the chip. A timing diagram for the load data command is shown in Figure 1.2.2.2.

1.2.2.3 Load Data for Data Memory

After receiving this command, the RB7 pin is set high, and the data can then be latched into the chip. When a 'Begin Programming' command is issued following this command, the data will be programmed into the data memory.

1.2.2.4 Read Data from Program Memory

After receiving this command, the chip will transmit data bits out of the program memory currently accessed (user or configuration) on the second rising edge of the clock input after pin RB7 has been set high. When the clock is taken high for the second time after RB7 is set high, the clock must remain high until all of the data has been read out. The high or low data words can be accessed while the clock is high by setting RB6 high or low, respectively. The pins RA<3:0>, and RB<5:0> go into output mode on the second rising clock edge, and they will revert back to input mode (hi-impedence) when the clock is set low. A timing diagram of this command is shown in Figure 1.2.2.3.

1.2.2.5 Read Data from Data Memory

After receiving this command, the chip will transmit data bits out of the currently accessed data memory location in a fashion identical to that of the read command for program memory.

1.2.2.6 Increment Address

The PC is incremented when this command is received.

FIGURE 1.2.2.1 - DATA IN/OUT FORMAT

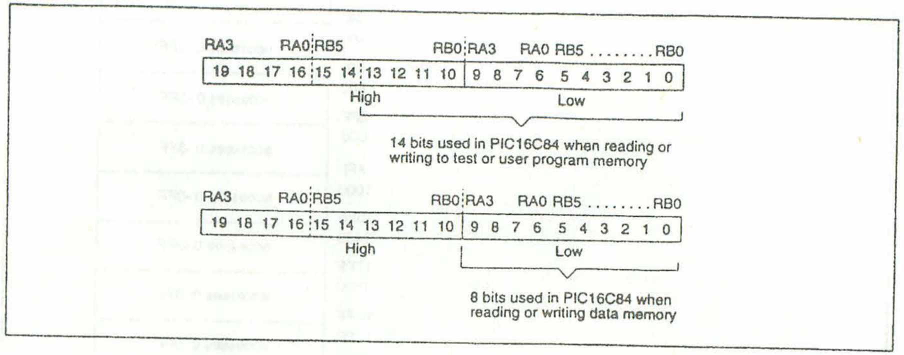


FIGURE 1.2.2.2 - LOAD DATA FOR PROGRAM MEMORY COMMAND (PARALLEL PROGRAM/ VERIFY)

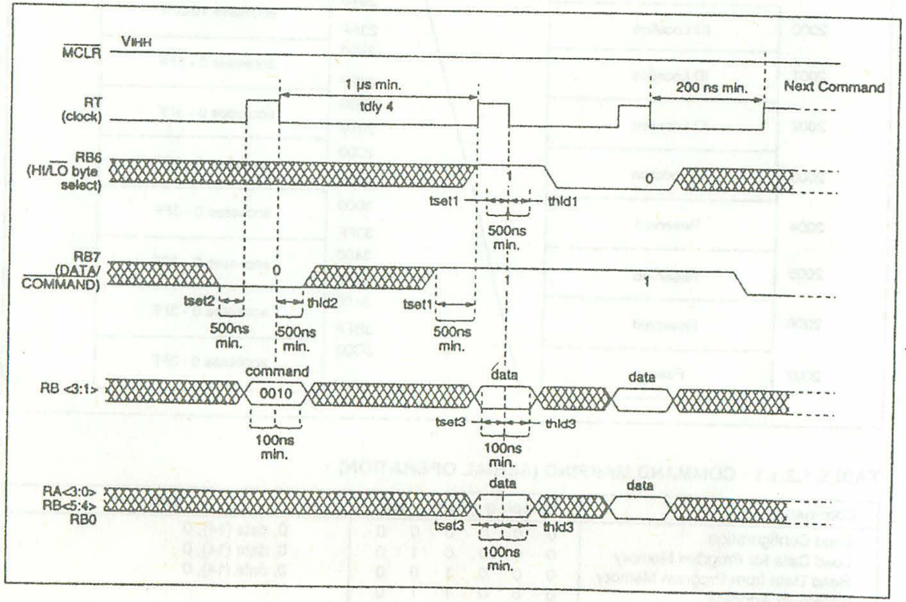


FIGURE 1.1.1 - PROGRAM MEMORY MAPPING

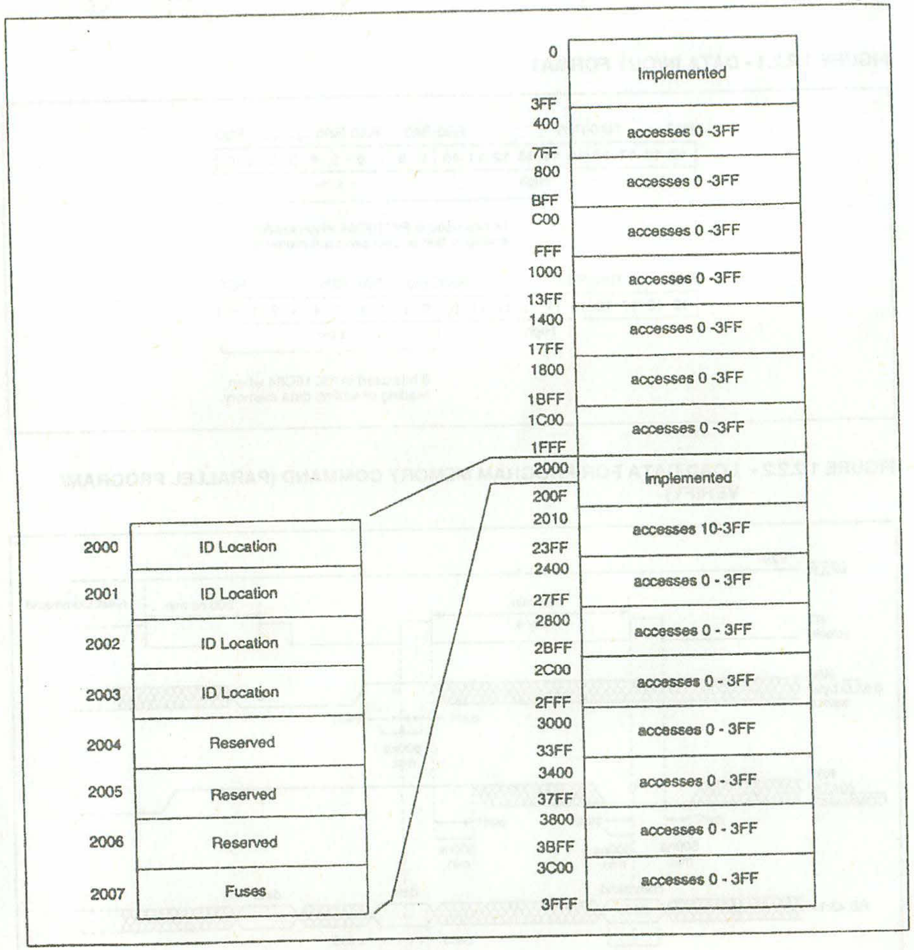
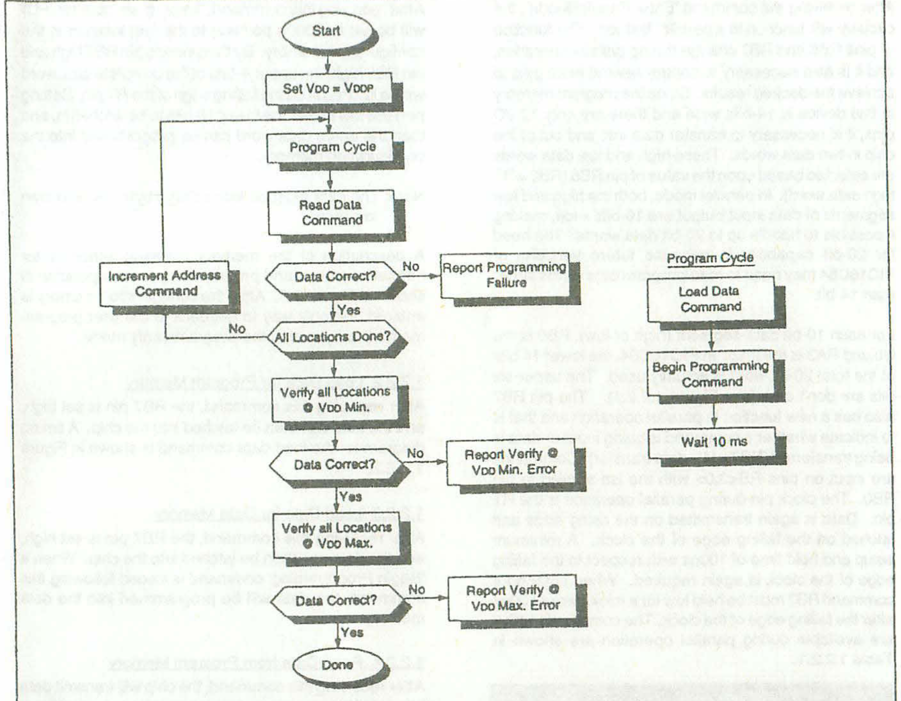


TABLE 1.2.1.1 - COMMAND MAPPING (SERIAL OPERATION)

Command	Mapping (msb ... lsb)						Data
Load Configuration	0	0	0	0	0	0	0, data (14), 0
Load Data for Program Memory	0	0	0	0	1	0	0, data (14), 0
Read Data from Program Memory	0	0	0	1	0	0	0, data (14), 0
Increment Address	0	0	0	1	1	0	
Begin Programming (Note 1)	0	0	1	0	0	0	
Enter Parallel Mode	0	0	1	0	1	0	
Load Data for Data Memory	0	0	0	0	1	1	0, data (14), 0
Read Data from Data Memory	0	0	0	1	0	1	0, data (14), 0

Notes: 1. Programming is self-timed, hence no end programming command.



VDDP = VDD range for programming (typically 4.5V - 5.5V)
VDD min. = Minimum VDD for device operation.
VDD max. = Maximum VDD for device operation.

Therefore, during a read operation the lsb will be transmitted onto pin RB7 on the rising edge of the second cycle, and during a load operation the lsb will be latched on the falling edge of the second cycle. A minimum 1us delay is also specified between consecutive commands.

All commands are transmitted lsb first. Data words are also transmitted lsb first. The data is transmitted on the rising edge and latched on the falling edge of the clock. To allow for decoding of commands and reversal of data pin configuration, a time separation of at least us is required between a command and a data word (or another command).

The commands that are available are:

1.2.1.1 Load Configuration

After receiving this command, the program counter (PC) will be set to 2000 hex. By then applying 16 cycles to the clock pin, the chip will load 14-bits in as the "data word", as described above, to be programmed into the configuration memory. A description of the memory mapping schemes of the program memory for normal operation and configuration mode operation is shown in figure 1.0. After the configuration memory is entered, the only way to get back to the user program memory is to exit the program/verify test mode by taking MCLR low (Vil).

1.2.1.2 Load Data for Program Memory

After receiving this command, the chip will load in 14-bits as a "data word" when 16 cycles are applied, as described previously. A timing diagram for the load data command is shown in Figure 1.2.1.3.